目錄

MQTT协议中文版

by mcxiaoke

最新版本: **v1.0.4 2017.04.05** (感谢 @hentaicreep 修复多处表格错位问题 PRs)

文档地址

- MQTT协议中文版
- PDF和ePub下载
- 中文翻译项目

概述

MQTT是一个客户端服务端架构的发布/订阅模式的消息传输协议。它的设计思想是轻巧、开放、简单、规范，易于实现。这些特点使得它对很多场景来说都是很好的选择，特别是对于受限的环境如机器与机器的通信（M2M）以及物联网环境（IoT）。

说明

> MQTT英文原版协议提供了Word格式和HTML格式，我翻译的时候用的Word文档，之前一直提供的是Word文档转换的HTML和PDF供浏览和下载，最近花时间整理了Markdown版本，可以更方便的分章节在线浏览了，转换为Markdown后部分表格的格式不太对，会逐步用图片代替。

目录

发现任何翻译问题或格式问题欢迎提**PR**帮忙完善。

旧版文档

> 已过期，建议使用GitBook版本 最新版本: v1.0.1 2015.10.22

文档	连接
中文版 HTML	MQTT 3.1.1 中文版
中文版 PDF	MQTT 3.1.1 中文版
英文版 HTML	MQTT Version 3.1.1
英文版 PDF	MQTT Version 3.1.1

许可协议

联系方式

- Blog: http://blog.mcxiaoke.com
- Github: https://github.com/mcxiaoke
- Email: github@mcxiaoke.com

开源项目

- Rx文档中文翻译: https://github.com/mcxiaoke/RxDocs
- MQTT协议中文版: https://github.com/mcxiaoke/mqtt
- Awesome-Kotlin: https://github.com/mcxiaoke/awesome-kotlin
- Kotlin-Koi: https://github.com/mcxiaoke/kotlin-koi
- Next公共组件库: https://github.com/mcxiaoke/Android-Next
- PackerNg极速打包: https://github.com/mcxiaoke/packer-ng-plugin
- Gradle渠道打包: https://github.com/mcxiaoke/gradle-packer-plugin
- EventBus实现xBus: https://github.com/mcxiaoke/xBus
- 蘑菇饭App: https://github.com/mcxiaoke/minicat
- 饭否客户端: https://github.com/mcxiaoke/fanfouapp-opensource
- Volley镜像: https://github.com/mcxiaoke/android-volley

第一章 概述 Introduction

1.1 MQTT协议的组织结构 Organization of MQTT

本规范分为七个章节：

1.2 术语 Terminology

本规范中用到的关键字 必须 MUST，不能 MUST NOT，要求 REQUIRED，将会 SHALL，不会 SHALL NOT，应该 SHOULD，不应该 SHOULD NOT，推荐 RECOMMENDED，可以 MAY，可选 OPTIONAL 都是按照 IETF RFC 2119 [RFC2119] 中的描述解释。

网络连接 **Network Connection**

MQTT使用的底层传输协议基础设施。

- 客户端使用它连接服务端。
- 它提供有序的、可靠的、双向字节流传输。

例子见4.2节。

应用消息 **Application Message** MQTT协议通过网络传输应用数据。应用消息通过MQTT传输时，它们有关联的服务质量（QoS）和主题（Topic）。

客户端 **Client**
使用MQTT的程序或设备。客户端总是通过网络连接到服务端。它可以

- 发布应用消息给其它相关的客户端。
- 订阅以请求接受相关的应用消息。
- 取消订阅以移除接受应用消息的请求。
- 从服务端断开连接。

服务端 **Server**
一个程序或设备，作为发送消息的客户端和请求订阅的客户端之间的中介。服务端

- 接受来自客户端的网络连接。
- 接受客户端发布的应用消息。
- 处理客户端的订阅和取消订阅请求。
- 转发应用消息给符合条件的已订阅客户端。

订阅 **Subscription**
订阅包含一个主题过滤器（Topic Filter）和一个最大的服务质量（QoS）等级。订阅与单个会话（Session）关联。会话可以包含多于一个的订阅。会话的每个订阅都有一个不同的主题过滤器。

主题名 **Topic Name**
附加在应用消息上的一个标签，服务端已知且与订阅匹配。服务端发送应用消息的一个副本给每一个匹配的客户端订阅。

主题过滤器 **Topic Filter**
订阅中包含的一个表达式，用于表示相关的一个或多个主题。主题过滤器可以使用通配符。

会话 **Session**
客户端和服务端之间的状态交互。一些会话持续时长与网络连接一样，另一些可以在客户端和服务端的多个连续网络连接间扩展。

控制报文 **MQTT Control Packet**
通过网络连接发送的信息数据包。MQTT规范定义了十四种不同类型的控制报文，其中一个（PUBLISH报文）用于传输应用消息。

1.5 数据表示 Data representations

1.5.1 二进制位 Bits

字节中的位从0到7。第7位是最高有效位，第0位是最低有效位。

1.5.2 整数数值 Integer data values

整数数值是16位，使用大端序（big-endian，高位字节在低位字节前面）。这意味着一个16位的字在网络上表示为最高有效字节（MSB），后面跟着最低有效字节（LSB）。

1.5.3 UTF-8编码字符串 UTF-8 encoded strings

后面会描述的控制报文中的文本字段编码为UTF-8格式的字符串。UTF-8 [RFC3629] 是一个高效的Unicode字符编码格式，为了支持基于文本的通信，它对ASCII字符的编码做了优化。

每一个字符串都有一个两字节的长度字段作为前缀，它给出这个字符串UTF-8编码的字节数，它们在图例 1.1 UTF-8编码字符串的结构 中描述。因此可以传送的UTF-8编码的字符串大小有一个限制，不能超过 65535字节。

除非另有说明，所有的UTF-8编码字符串的长度都必须在0到65535字节这个范围内。

图例 **1.1 UTF-8**编码字符串的结构 **Structure of UTF-8 encoded strings**

二进制位	**7-0**
byte 1	字符串长度的最高有效字节（MSB）
byte 2	字符串长度的最低有效字节（LSB）
byte 3 ….	如果长度大于0，这里是UTF-8编码的字符数据。

UTF-8编码字符串中的字符数据必须是按照Unicode规范 [Unicode] 定义的和在RFC3629 [RFC3629] 中重申的有效的UTF-8格式。特别需要指出的是，这些数据不能包含字符码在U+D800和U+DFFF之间的数据。如果服务端或客户端收到了一个包含无效UTF-8字符的控制报文，它必须关闭网络连接 [MQTT-1.5.3-1]。

UTF-8编码的字符串不能包含空字符U+0000。如果客户端或服务端收到了一个包含U+0000的控制报文，它必须关闭网络连接 [MQTT-1.5.3-2]。

数据中不应该包含下面这些Unicode代码点的编码。如果一个接收者（服务端或客户端）收到了包含下列任意字符的控制报文，它可以关闭网络连接：

- U+0001和U+001F之间的控制字符
- U+007F和U+009F之间的控制字符
- Unicode规范定义的非字符代码点（例如U+0FFFF）
- Unicode规范定义的保留字符（例如U+0FFFF）

UTF-8编码序列0XEF 0xBB 0xBF总是被解释为U+FEFF（零宽度非换行空白字符），无论它出现在字符串的什么位置，报文接收者都不能跳过或者剥离它 [MQTT-1.5.3-3]。

非规范示例 Non normative example

> 例如，字符串 A　是一个拉丁字母A后面跟着一个代码点U+2A6D4(它表示一个中日韩统一表意文字扩展B中的字符)，这个字符串编码如下：

图例 **1.2 UTF-8**编码字符串非规范示例 **UTF-8 encoded string non normative example**

Bit	7	6	5	4	3	2	1	0
byte 1	字符串长度 MSB (0x00)							
	0	0	0	0	0	0	0	0
byte 2	字符串长度 LSB (0x05)							
	0	0	0	0	0	1	0	1
byte 3	‘A’ (0x41)							
	0	1	0	0	0	0	0	1
byte 4	(0xF0)							
	1	1	1	1	0	0	0	0
byte 5	(0xAA)							
	1	0	1	0	1	0	1	0
byte 6	(0x9B)							
	1	0	0	1	1	0	1	1
byte 7	(0x94)							
	1	0	0	1	0	1	0	0

1.6 编辑约定 Editing conventions

本规范用黄色高亮的文本标识一致性声明，每个一致性声明都分配了一个这种格式的引用：[MQTT-x.x.x-y]。

项目主页

- MQTT协议中文版

第二章 MQTT控制报文格式 MQTT Control Packet format

目录

2.1 MQTT控制报文的结构 Structure of an MQTT Control Packet

MQTT协议通过交换预定义的MQTT控制报文来通信。这一节描述这些报文的格式。

MQTT控制报文由三部分组成，按照 图例 2.1 –MQTT控制报文的结构 描述的顺序：

图例 **2.1 –MQTT**控制报文的结构

Fixed header	固定报头，所有控制报文都包含
Variable header	可变报头，部分控制报文包含
Payload	有效载荷，部分控制报文包含

2.2 固定报头 Fixed header

每个MQTT控制报文都包含一个固定报头。图例 2.2 -固定报头的格式 描述了固定报头的格式。

图例 **2.2 -**固定报头的格式

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT控制报文的类型				用于指定控制报文类型的标志位			
byte 2...	剩余长度							

2.2.1 MQTT控制报文的类型 MQTT Control Packet type

位置：第1个字节，二进制位7-4。

表示为4位无符号值，这些值的定义见 表格 2.1 -控制报文的类型

表格 **2.1 -**控制报文的类型

名字	值	报文流动方向	描述
Reserved	0	禁止	保留
CONNECT	1	客户端到服务端	客户端请求连接服务端
CONNACK	2	服务端到客户端	连接报文确认
PUBLISH	3	两个方向都允许	发布消息
PUBACK	4	两个方向都允许	QoS 1消息发布收到确认
PUBREC	5	两个方向都允许	发布收到（保证交付第一步）
PUBREL	6	两个方向都允许	发布释放（保证交付第二步）
PUBCOMP	7	两个方向都允许	QoS 2消息发布完成（保证交互第三步）
SUBSCRIBE	8	客户端到服务端	客户端订阅请求
SUBACK	9	服务端到客户端	订阅请求报文确认
UNSUBSCRIBE	10	客户端到服务端	客户端取消订阅请求
UNSUBACK	11	服务端到客户端	取消订阅报文确认
PINGREQ	12	客户端到服务端	心跳请求
PINGRESP	13	服务端到客户端	心跳响应
DISCONNECT	14	客户端到服务端	客户端断开连接
Reserved	15	禁止	保留

2.2.2 标志 Flags

固定报头第1个字节的剩余的4位 [3-0]包含每个MQTT控制报文类型特定的标志，见 表格 2.2 -标志位。表格 2.2中任何标记为“保留”的标志位，都是保留给以后使用的，必须设置为表格中列出的值 [MQTT-2.2.2-1]。如果收到非法的标志，接收者必须关闭网络连接。有关错误处理的详细信息见 4.8节 [MQTT-2.2.2-2]。

表格 **2.2 -** 标志位 **Flag Bits**

控制报文	固定报头标志	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
CONNECT	Reserved	0	0	0	0
CONNACK	Reserved	0	0	0	0
PUBLISH	Used in MQTT 3.1.1	DUP[1]	QoS[2]	QoS[2]	RETAIN[3]
PUBACK	Reserved	0	0	0	0
PUBREC	Reserved	0	0	0	0
PUBREL	Reserved	0	0	1	0
PUBCOMP	Reserved	0	0	0	0
SUBSCRIBE	Reserved	0	0	1	0
SUBACK	Reserved	0	0	0	0
UNSUBSCRIBE	Reserved	0	0	1	0
UNSUBACK	Reserved	0	0	0	0
PINGREQ	Reserved	0	0	0	0
PINGRESP	Reserved	0	0	0	0
DISCONNECT	Reserved	0	0	0	0

- DUP[1] =控制报文的重复分发标志
- QoS[2] = PUBLISH报文的服务质量等级
- RETAIN[3] = PUBLISH报文的保留标志

PUBLISH控制报文中的DUP, QoS和RETAIN标志的描述见 3.3.1节。

2.2.3 剩余长度 Remaining Length

位置：从第2个字节开始。

剩余长度（Remaining Length）表示当前报文剩余部分的字节数，包括可变报头和负载的数据。剩余长度不包括用于编码剩余长度字段本身的字节数。

剩余长度字段使用一个变长度编码方案，对小于128的值它使用单字节编码。更大的值按下面的方式处理。低7位有效位用于编码数据，最高有效位用于指示是否有更多的字节。因此每个字节可以编码128个数值和一个延续位（*continuation bit*）。剩余长度字段最大4个字节。

> 非规范评注
>
> 例如，十进制数64会被编码为一个字节，数值是64，十六进制表示为0x40,。十进制数字321(=65+2*128)被编码为两个字节，最低有效位在前。第一个字节是 65+128=193。注意最高位为1表示后面至少还有一个字节。第二个字节是2。
>
> 非规范评注
>
> 这允许应用发送最大256MB(268,435,455)大小的控制报文。这个数值在报文中的表示是：0xFF,0xFF,0xFF,0x7F。
>
> 表格 2.4剩余长度字段的大小展示了剩余长度字段所表示的值随字节增长。

表格 **2.4**剩余长度字段的大小 **Size of Remaining Length field**

字节数	最小值	最大值
1	0 (0x00)	127 (0x7F)
2	128 (0x80, 0x01)	16 383 (0xFF, 0x7F)
3	16 384 (0x80, 0x80, 0x01)	2 097 151 (0xFF, 0xFF, 0x7F)
4	2 097 152 (0x80, 0x80, 0x80, 0x01)	268 435 455 (0xFF, 0xFF, 0xFF, 0x7F)

分别表示（每个字节的低7位用于编码数据，最高位是标志位）：

- 1个字节时，从0(0x00)到127(0x7f)
- 2个字节时，从128(0x80,0x01)到16383(0Xff,0x7f)
- 3个字节时，从16384(0x80,0x80,0x01)到2097151(0xFF,0xFF,0x7F)
- 4个字节时，从2097152(0x80,0x80,0x80,0x01)到268435455(0xFF,0xFF,0xFF,0x7F)

> 非规范评注
>
> 非负整数X使用变长编码方案的算法如下：
>
> ```
>     do
>    encodedByte = X MOD 128
>    X = X DIV 128
>   // if there are more data to encode, set the top bit of this byte
>   if ( X > 0 )
>       encodedByte = encodedByte OR 128
>   endif
>       'output' encodedByte
>  while ( X > 0 )
> ```
>
> MOD是模运算，DIV是整数除法，OR是位操作或（C语言中分别是%，/，|）
>
> 非规范评注
>
> 剩余长度字段的解码算法如下：
>
> ```
> multiplier = 1
> value = 0
> do
>     encodedByte = 'next byte from stream'
>     value += (encodedByte AND 127) * multiplier
>     multiplier *= 128
>     if (multiplier > 128*128*128)
>        throw Error(Malformed Remaining Length)
> while ((encodedByte AND 128) != 0)
> ```
>
> AND是位操作与（C语言中的&）
>
> 这个算法终止时，value包含的就是剩余长度的值。

2.3 可变报头 Variable header

某些MQTT控制报文包含一个可变报头部分。它在固定报头和负载之间。可变报头的内容根据报文类型的不同而不同。可变报头的报文标识符（Packet Identifier）字段存在于在多个类型的报文里。

2.3.1 报文标识符 Packet Identifier

图例 **2.3 -**报文标识符字节 **Packet Identifier bytes**

Bit	7 - 0
byte 1	报文标识符 MSB
byte 2	报文标识符 LSB

很多控制报文的可变报头部分包含一个两字节的报文标识符字段。这些报文是PUBLISH（QoS > 0时），PUBACK，PUBREC，PUBREL，PUBCOMP，SUBSCRIBE, SUBACK，UNSUBSCIBE，UNSUBACK。

SUBSCRIBE，UNSUBSCRIBE和PUBLISH（QoS大于0）控制报文必须包含一个非零的16位报文标识符（Packet Identifier）[MQTT-2.3.1-1]。客户端每次发送一个新的这些类型的报文时都必须分配一个当前未使用的报文标识符 [MQTT-2.3.1-2]。如果一个客户端要重发这个特殊的控制报文，在随后重发那个报文时，它必须使用相同的标识符。当客户端处理完这个报文对应的确认后，这个报文标识符就释放可重用。QoS 1的PUBLISH对应的是PUBACK，QoS 2的PUBLISH对应的是PUBCOMP，与SUBSCRIBE或UNSUBSCRIBE对应的分别是SUBACK或UNSUBACK [MQTT-2.3.1-3]。发送一个QoS 0的PUBLISH报文时，相同的条件也适用于服务端 [MQTT-2.3.1-4]。

QoS等于0的PUBLISH报文不能包含报文标识符 [MQTT-2.3.1-5]。

PUBACK, PUBREC, PUBREL报文必须包含与最初发送的PUBLISH报文相同的报文标识符 [MQTT-2.3.1-6]。类似地，SUBACK和UNSUBACK必须包含在对应的SUBSCRIBE和UNSUBSCRIBE报文中使用的报文标识符 [MQTT-2.3.1-7]。

需要报文标识符的控制报文在 表格 2.5 -包含报文标识符的控制报文 中列出。

表格 **2.5 -**包含报文标识符的控制报文 **Control Packets that contain a Packet Identifier**

控制报文	报文标识符字段
CONNECT	不需要
CONNACK	不需要
PUBLISH	需要（如果QoS > 0）
PUBACK	需要
PUBREC	需要
PUBREL	需要
PUBCOMP	需要
SUBSCRIBE	需要
SUBACK	需要
UNSUBSCRIBE	需要
UNSUBACK	需要
PINGREQ	不需要
PINGRESP	不需要
DISCONNECT	不需要

客户端和服务端彼此独立地分配报文标识符。因此，客户端服务端组合使用相同的报文标识符可以实现并发的消息交换。

非规范评注

客户端发送标识符为0x1234的PUBLISH报文，它有可能会在收到那个报文的PUBACK之前，先收到服务端发送的另一个不同的但是报文标识符也为0x1234的PUBLISH报文。

Client	**Server**
PUBLISH	Packet Identifier=0x1234---
--PUBLISH	Packet Identifier=0x1234
PUBACK	Packet Identifier=0x1234---
--PUBACK	Packet Identifier=0x1234

2.4 有效载荷 Payload

某些MQTT控制报文在报文的最后部分包含一个有效载荷，这将在第三章论述。对于PUBLISH来说有效载荷就是应用消息。表格 2.6 – 包含有效载荷的控制报文 列出了需要有效载荷的控制报文。

表格 2.6 – 包含有效载荷的控制报文 Control Packets that contain a Payload

控制报文	有效载荷
CONNECT	需要
CONNACK	不需要
PUBLISH	可选
PUBACK	不需要
PUBREC	不需要
PUBREL	不需要
PUBCOMP	不需要
SUBSCRIBE	需要
SUBACK	需要
UNSUBSCRIBE	需要
UNSUBACK	不需要
PINGREQ	不需要
PINGRESP	不需要
DISCONNECT	不需要

项目主页

- MQTT协议中文版

第三章 MQTT控制报文 MQTT Control Packets

目录

本章目录

项目主页

- MQTT协议中文版

3.1 CONNECT – 连接服务端

客户端到服务端的网络连接建立后，客户端发送给服务端的第一个报文必须是CONNECT报文 [MQTT-3.1.0-1]。

在一个网络连接上，客户端只能发送一次CONNECT报文。服务端必须将客户端发送的第二个CONNECT报文当作协议违规处理并断开客户端的连接 [MQTT-3.1.0-2]。有关错误处理的信息请查看4.8节。

有效载荷包含一个或多个编码的字段。包括客户端的唯一标识符，Will主题，Will消息，用户名和密码。除了客户端标识之外，其它的字段都是可选的，基于标志位来决定可变报头中是否需要包含这些字段。

3.1.1 固定报头 Fixed header

图例 **3.1 –CONNECT**报文的固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT报文类型 (1)				Reserved 保留位			
	0	0	0	1	0	0	0	0
byte 2...	剩余长度							

剩余长度字段

剩余长度等于可变报头的长度（10字节）加上有效载荷的长度。编码方式见2.2.3节的说明。

3.1.2 可变报头 Variable header

CONNECT报文的可变报头按下列次序包含四个字段：协议名（Protocol Name），协议级别（Protocol Level），连接标志（Connect Flags）和保持连接（Keep Alive）。

协议名 Protocol Name

图例 **3.2 -**协议名字节构成

	说明	7	6	5	4	3	2	1	0
协议名									
byte 1	长度 MSB (0)	0	0	0	0	0	0	0	0
byte 2	长度 LSB (4)	0	0	0	0	0	1	0	0
byte 3	‘M’	0	1	0	0	1	1	0	1
byte 4	‘Q’	0	1	0	1	0	0	0	1
byte 5	‘T’	0	1	0	1	0	1	0	0
byte 6	‘T’	0	1	0	1	0	1	0	0

协议名是表示协议名 *MQTT* 的UTF-8编码的字符串。MQTT规范的后续版本不会改变这个字符串的偏移和长度。

如果协议名不正确服务端可以断开客户端的连接，也可以按照某些其它规范继续处理CONNECT报文。对于后一种情况，按照本规范，服务端不能继续处理CONNECT报文 [MQTT-3.1.2-1]。

> 非规范评注
>
> 数据包检测工具，例如防火墙，可以使用协议名来识别MQTT流量。

协议级别 Protocol Level

图例 **3.3 - Protocol Level byte**协议级别字节构成

	说明	7	6	5	4	3	2	1	0
协议级别									
byte 7	Level(4)	0	0	0	0	0	1	0	0

客户端用8位的无符号值表示协议的修订版本。对于3.1.1版协议，协议级别字段的值是4(0x04)。如果发现不支持的协议级别，服务端必须给发送一个返回码为0x01（不支持的协议级别）的CONNACK报文响应CONNECT报文，然后断开客户端的连接 [MQTT-3.1.2-2]。

连接标志 Connect Flags

连接标志字节包含一些用于指定MQTT连接行为的参数。它还指出有效载荷中的字段是否存在。

图例 **3.4 -**连接标志位

Bit	7	6	5	4	3	2	1	0
	User Name Flag	Password Flag	Will Retain	Will QoS		Will Flag	Clean Session	Reserved
byte 8	X	X	X	X	X	X	X	0

服务端必须验证CONNECT控制报文的保留标志位（第0位）是否为0，如果不为0必须断开客户端连接 [MQTT-3.1.2-3]。

清理会话 Clean Session

位置：连接标志字节的第1位

这个二进制位指定了会话状态的处理方式。

客户端和服务端可以保存会话状态，以支持跨网络连接的可靠消息传输。这个标志位用于控制会话状态的生存时间。

如果清理会话（CleanSession）标志被设置为0，服务端必须基于当前会话（使用客户端标识符识别）的状态恢复与客户端的通信。如果没有与这个客户端标识符关联的会话，服务端必须创建一个新的会话。在连接断开之后，当连接断开后，客户端和服务端必须保存会话信息 [MQTT-3.1.2-4]。当清理会话标志为0的会话连接断开之后，服务端必须将之后的QoS 1和QoS 2级别的消息保存为会话状态的一部分，如果这些消息匹配断开连接时客户端的任何订阅 [MQTT-3.1.2-5]。服务端也可以保存满足相同条件的QoS 0级别的消息。

如果清理会话（CleanSession）标志被设置为1，客户端和服务端必须丢弃之前的任何会话并开始一个新的会话。会话仅持续和网络连接同样长的时间。与这个会话关联的状态数据不能被任何之后的会话重用 [MQTT-3.1.2-6]。

客户端的会话状态包括：

- 已经发送给服务端，但是还没有完成确认的QoS 1和QoS 2级别的消息
- 已从服务端接收，但是还没有完成确认的QoS 2级别的消息。

服务端的会话状态包括：

- 会话是否存在，即使会话状态的其它部分都是空。
- 客户端的订阅信息。
- 已经发送给客户端，但是还没有完成确认的QoS 1和QoS 2级别的消息。
- 即将传输给客户端的QoS 1和QoS 2级别的消息。
- 已从客户端接收，但是还没有完成确认的QoS 2级别的消息。
- 可选，准备发送给客户端的QoS 0级别的消息。

保留消息不是服务端会话状态的一部分，会话终止时不能删除保留消息 [MQTT-3.1.2.7]。

有关状态存储的限制和细节见第 4.1节。

当清理会话标志被设置为1时，客户端和服务端的状态删除不需要是原子操作。

> 非规范评注
>
> 为了确保在发生故障时状态的一致性，客户端应该使用会话状态标志1重复请求连接，直到连接成功。
>
> 非规范评注
>
> 一般来说，客户端连接时总是将清理会话标志设置为0或1，并且不交替使用两种值。这个选择取决于具体的应用。清理会话标志设置为1的客户端不会收到旧的应用消息，而且在每次连接成功后都需要重新订阅任何相关的主题。清理会话标志设置为0的客户端会收到所有在它连接断开期间发布的QoS 1和QoS 2级别的消息。因此，要确保不丢失连接断开期间的消息，需要使用QoS 1或 QoS 2级别，同时将清理会话标志设置为0。
>
> 非规范评注
>
> 清理会话标志0的客户端连接时，它请求服务端在连接断开后保留它的MQTT会话状态。如果打算在之后的某个时间点重连到这个服务端，客户端连接应该只使用清理会话标志0。当客户端决定之后不再使用这个会话时，应该将清理会话标志设置为1最后再连接一次，然后断开连接。

遗嘱标志 Will Flag

位置：连接标志的第2位。

遗嘱标志（Will Flag）被设置为1，表示如果连接请求被接受了，遗嘱（Will Message）消息必须被存储在服务端并且与这个网络连接关联。之后网络连接关闭时，服务端必须发布这个遗嘱消息，除非服务端收到DISCONNECT报文时删除了这个遗嘱消息 [MQTT-3.1.2-8] 。

遗嘱消息发布的条件，包括但不限于：

- 服务端检测到了一个I/O错误或者网络故障。
- 客户端在保持连接（Keep Alive）的时间内未能通讯。
- 客户端没有先发送DISCONNECT报文直接关闭了网络连接。
- 由于协议错误服务端关闭了网络连接。

如果遗嘱标志被设置为1，连接标志中的Will QoS和Will Retain字段会被服务端用到，同时有效载荷中必须包含Will Topic和Will Message字段 [MQTT-3.1.2-9]。

一旦被发布或者服务端收到了客户端发送的DISCONNECT报文，遗嘱消息就必须从存储的会话状态中移除 [MQTT-3.1.2-10]。

如果遗嘱标志被设置为0，连接标志中的Will QoS和Will Retain字段必须设置为0，并且有效载荷中不能包含Will Topic和Will Message字段 [MQTT-3.1.2-11]。

如果遗嘱标志被设置为0，网络连接断开时，不能发送遗嘱消息 [MQTT-3.1.2-12]。

服务端应该迅速发布遗嘱消息。在关机或故障的情况下，服务端可以推迟遗嘱消息的发布直到之后的重启。如果发生了这种情况，在服务器故障和遗嘱消息被发布之间可能会有一个延迟。

遗嘱QoS Will QoS

位置：连接标志的第4和第3位。

这两位用于指定发布遗嘱消息时使用的服务质量等级。

如果遗嘱标志被设置为0，遗嘱QoS也必须设置为0(0x00) [MQTT-3.1.2-13]。

如果遗嘱标志被设置为1，遗嘱QoS的值可以等于0(0x00)，1(0x01)，2(0x02)。它的值不能等于3 [MQTT-3.1.2-14]。

遗嘱保留 Will Retain

位置：连接标志的第5位。

如果遗嘱消息被发布时需要保留，需要指定这一位的值。

如果遗嘱标志被设置为0，遗嘱保留（Will Retain）标志也必须设置为0 [MQTT-3.1.2-15]。

如果遗嘱标志被设置为1：

- 如果遗嘱保留被设置为0，服务端必须将遗嘱消息当作非保留消息发布 [MQTT-3.1.2-16]。
- 如果遗嘱保留被设置为1，服务端必须将遗嘱消息当作保留消息发布 [MQTT-3.1.2-17]。

用户名标志 User Name Flag

位置：连接标志的第7位。

如果用户名（User Name）标志被设置为0，有效载荷中不能包含用户名字段 [MQTT-3.1.2-18]。

如果用户名（User Name）标志被设置为1，有效载荷中必须包含用户名字段 [MQTT-3.1.2-19]。

密码标志 Password Flag

位置：连接标志的第6位。

如果密码（Password）标志被设置为0，有效载荷中不能包含密码字段 [MQTT-3.1.2-20]。

如果密码（Password）标志被设置为1，有效载荷中必须包含密码字段 [MQTT-3.1.2-21]。

如果用户名标志被设置为0，密码标志也必须设置为0 [MQTT-3.1.2-22]。

保持连接 Keep Alive

图例 **3.5**保持连接字节

Bit	7	6	5	4	3	2	1	0
byte 9	保持连接 Keep Alive MSB							
byte 10	保持连接 Keep Alive LSB							

保持连接（Keep Alive）是一个以秒为单位的时间间隔，表示为一个16位的字，它是指在客户端传输完成一个控制报文的时刻到发送下一个报文的时刻，两者之间允许空闲的最大时间间隔。客户端负责保证控制报文发送的时间间隔不超过保持连接的值。如果没有任何其它的控制报文可以发送，客户端必须发送一个PINGREQ报文 [MQTT-3.1.2-23]。

不管保持连接的值是多少，客户端任何时候都可以发送PINGREQ报文，并且使用PINGRESP报文判断网络和服务端的活动状态。

如果保持连接的值非零，并且服务端在一点五倍的保持连接时间内没有收到客户端的控制报文，它必须断开客户端的网络连接，认为网络连接已断开 [MQTT-3.1.2-24]。

客户端发送了PINGREQ报文之后，如果在合理的时间内仍没有收到PINGRESP报文，它应该关闭到服务端的网络连接。

保持连接的值为零表示关闭保持连接功能。这意味着，服务端不需要因为客户端不活跃而断开连接。注意：不管保持连接的值是多少，任何时候，只要服务端认为客户端是不活跃或无响应的，可以断开客户端的连接。

> 非规范评注
>
> 保持连接的实际值是由应用指定的，一般是几分钟。允许的最大值是18小时12分15秒。

可变报头非规范示例

图例 **3.6 -**可变报头非规范示例

Figure 3.6 - Variable header non normative example

	Description	7	6	5	4	3	2	1	0
Protocol Name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (4)	0	0	0	0	0	1	0	0
byte 3	'M'	0	1	0	0	1	1	0	1
byte 4	'Q'	0	1	0	1	0	0	0	1
byte 5	'T'	0	1	0	1	0	1	0	0
byte 6	'T'	0	1	0	1	0	1	0	0
Protocol Level									
	Description	7	6	5	4	3	2	1	0
byte 7	Level (4)	0	0	0	0	0	1	0	0
Connect Flags									
byte 8	User Name Flag (1) Password Flag (1) Will Retain (0) Will QoS (01) Will Flag (1) Clean Session (1) *Reserved* (0)	1	1	0	0	1	1	1	0
Keep Alive									
byte 9	Keep Alive MSB (0)	0	0	0	0	0	0	0	0
byte 10	Keep Alive LSB (10)	0	0	0	0	1	0	1	0

3.1.3 有效载荷 Payload

CONNECT报文的有效载荷（payload）包含一个或多个以长度为前缀的字段，可变报头中的标志决定是否包含这些字段。如果包含的话，必须按这个顺序出现：客户端标识符，遗嘱主题，遗嘱消息，用户名，密码 [MQTT-3.1.3-1]。

客户端标识符 Client Identifier

服务端使用客户端标识符 (ClientId) 识别客户端。连接服务端的每个客户端都有唯一的客户端标识符（ClientId）。客户端和服务端都必须使用ClientId识别两者之间的MQTT会话相关的状态 [MQTT-3.1.3-2]。

客户端标识符 (ClientId) 必须存在而且必须是CONNECT报文有效载荷的第一个字段 [MQTT-3.1.3-3]。

客户端标识符必须是1.5.3节定义的UTF-8编码字符串 [MQTT-3.1.3-4]。

服务端必须允许1到23个字节长的UTF-8编码的客户端标识符，客户端标识符只能包含这些字符："0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"（大写字母，小写字母和数字）[MQTT-3.1.3-5]。

服务端可以允许编码后超过23个字节的客户端标识符 (ClientId)。服务端可以允许包含不是上面列表字符的客户端标识符 (ClientId)。

服务端可以允许客户端提供一个零字节的客户端标识符 (ClientId) ，如果这样做了，服务端必须将这看作特殊情况并分配唯一的客户端标识符给那个客户端。然后它必须假设客户端提供了那个唯一的客户端标识符，正常处理这个CONNECT报文 [MQTT-3.1.3-6]。

如果客户端提供了一个零字节的客户端标识符，它必须同时将清理会话标志设置为1 [MQTT-3.1.3-7]。

如果客户端提供的ClientId为零字节且清理会话标志为0，服务端必须发送返回码为0x02（表示标识符不合格）的CONNACK报文响应客户端的CONNECT报文，然后关闭网络连接 [MQTT-3.1.3-8]。

如果服务端拒绝了这个ClientId，它必须发送返回码为0x02（表示标识符不合格）的CONNACK报文响应客户端的CONNECT报文，然后关闭网络连接 [MQTT-3.1.3-9]。

> 非规范评注
>
> 客户端实现可以提供一个方便的方法用于生成随机的ClientId。当清理会话标志被设置为0时应该主动放弃使用这种方法。

遗嘱主题 Will Topic

如果遗嘱标志被设置为1，有效载荷的下一个字段是遗嘱主题（Will Topic）。遗嘱主题必须是1.5.3节定义的UTF-8编码字符串 [MQTT-3.1.3-10]。

遗嘱消息 Will Message

如果遗嘱标志被设置为1，有效载荷的下一个字段是遗嘱消息。遗嘱消息定义了将被发布到遗嘱主题的应用消息，见3.1.2.5节的描述。这个字段由一个两字节的长度和遗嘱消息的有效载荷组成，表示为零字节或多个字节序列。长度给出了跟在后面的数据的字节数，不包含长度字段本身占用的两个字节。

遗嘱消息被发布到遗嘱主题时，它的有效载荷只包含这个字段的数据部分，不包含开头的两个长度字节。

用户名 User Name

如果用户名（User Name）标志被设置为1，有效载荷的下一个字段就是它。用户名必须是1.5.3节定义的UTF-8编码字符串 [MQTT-3.1.3-11]。服务端可以将它用于身份验证和授权。

密码 Password

如果密码（Password）标志被设置为1，有效载荷的下一个字段就是它。密码字段包含一个两字节的长度字段，长度表示二进制数据的字节数（不包含长度字段本身占用的两个字节），后面跟着0到65535字节的二进制数据。

图例 **3.7 -** 密码字节

Bit	7 - 0
byte 1	数据长度 MSB
byte 2	数据长度 LSB
byte 3 ….	如果长度大于0，这里就是数据部分

3.1.4 响应 Response

注意：服务器可以在同一个TCP端口或其他网络端点上支持多种协议（包括本协议的早期版本）。如果服务器确定协议是MQTT 3.1.1，那么它按照下面的方法验证连接请求。

1. 网络连接建立后，如果服务端在合理的时间内没有收到CONNECT报文，服务端应该关闭这个连接。
2. 服务端必须按照3.1节的要求验证CONNECT报文，如果报文不符合规范，服务端不发送CONNACK报文直接关闭网络连接 [MQTT-3.1.4-1]。
3. 服务端可以检查CONNECT报文的内容是不是满足任何进一步的限制，可以执行身份验证和授权检查。如果任何一项检查没通过，按照3.2节的描述，它应该发送一个适当的、返回码非零的CONNACK响应，并且必须关闭这个网络连接。

如果验证成功，服务端会执行下列步骤。

1. 如果ClientId表明客户端已经连接到这个服务端，那么服务端必须断开原有的客户端连接 [MQTT-3.1.4-2]。
2. 服务端必须按照 3.1.2.4节的描述执行清理会话的过程 [MQTT-3.1.4-3]。
3. 服务端必须发送返回码为零的CONNACK报文作为CONNECT报文的确认响应 [MQTT-3.1.4-4]。
4. 开始消息分发和保持连接状态监视。

允许客户端在发送CONNECT报文之后立即发送其它的控制报文；客户端不需要等待服务端的CONNACK报文。如果服务端拒绝了CONNECT，它不能处理客户端在CONNECT报文之后发送的任何数据 [MQTT-3.1.4-5]。

> 非规范评注
>
> 客户端通常会等待一个CONNACK报文。然而客户端有权在收到CONNACK之前发送控制报文，由于不需要维持连接状态，这可以简化客户端的实现。

第三章目录 **MQTT**控制报文

- 3.0 Contents – MQTT控制报文
- 3.1 CONNECT – 连接服务端
- 3.2 CONNACK – 确认连接请求
- 3.3 PUBLISH – 发布消息
- 3.4 PUBACK –发布确认
- 3.5 PUBREC – 发布收到（QoS 2，第一步）
- 3.6 PUBREL – 发布释放（QoS 2，第二步）
- 3.7 PUBCOMP – 发布完成（QoS 2，第三步）
- 3.8 SUBSCRIBE - 订阅主题
- 3.9 SUBACK – 订阅确认
- 3.10 UNSUBSCRIBE –取消订阅
- 3.11 UNSUBACK – 取消订阅确认
- 3.12 PINGREQ – 心跳请求
- 3.13 PINGRESP – 心跳响应
- 3.14 DISCONNECT –断开连接

项目主页

- MQTT协议中文版

3.2 CONNACK – 确认连接请求

服务端发送CONNACK报文响应从客户端收到的CONNECT报文。服务端发送给客户端的第一个报文必须是CONNACK [MQTT-3.2.0-1]。

如果客户端在合理的时间内没有收到服务端的CONNACK报文，客户端应该关闭网络连接。合理 的时间取决于应用的类型和通信基础设施。

3.2.1 固定报头

固定报头的格式见 图例 3.8 – CONNACK 报文固定报头 的描述。

图例 **3.8 – CONNACK** 报文固定报头

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 1	MQTT报文类型 (2)				Reserved 保留位			
	0	0	1	0	0	0	0	0
byte 2...	剩余长度 (2)							
	0	0	0	0	0	0	1	0

剩余长度字段

表示可变报头的长度。对于CONNACK报文这个值等于2。

3.2.2 可变报头

可变报头的格式见 图例 3.9 –CONNACK报文可变报头 的描述。

图例 **3.9 –CONNACK**报文可变报头

	描述	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
连接确认标志	Reserved 保留位	SP[1]							
byte 1		0	0	0	0	0	0	0	X
连接返回码									
byte 2		X	X	X	X	X	X	X	X

	描述	7	6	5	4	3	2	1	0
连接确认标志		Reserved 保留位							SP[1]
byte 1		0	0	0	0	0	0	0	X
连接返回码									
byte 2		X	X	X	X	X	X	X	X

连接确认标志 Connect Acknowledge Flags

第1个字节是 连接确认标志，位7-1是保留位且必须设置为0。 第0 (SP)位 是当前会话（Session Present）标志。

当前会话 Session Present

位置：连接确认标志的第0位。

如果服务端收到清理会话（CleanSession）标志为1的连接，除了将CONNACK报文中的返回码设置为0之外，还必须将CONNACK报文中的当前会话设置（Session Present）标志为0 [MQTT-3.2.2-1]。

如果服务端收到一个CleanSession为0的连接，当前会话标志的值取决于服务端是否已经保存了ClientId对应客户端的会话状态。如果服务端已经保存了会话状态，它必须将CONNACK报文中的当前会话标志设置为1 [MQTT-3.2.2-2]。如果服务端没有已保存的会话状态，它必须将CONNACK报文中的当前会话设置为0。还需要将CONNACK报文中的返回码设置为0 [MQTT-3.2.2-3]。

当前会话标志使服务端和客户端在是否有已存储的会话状态上保持一致。

一旦完成了会话的初始化设置，已经保存会话状态的客户端将期望服务端维持它存储的会话状态。如果客户端从服务端收到的当前的值与预期的不同，客户端可以选择继续这个会话或者断开连接。客户端可以丢弃客户端和服务端之间的会话状态，方法是，断开连接，将清理会话标志设置为1，再次连接，然后再次断开连接。

如果服务端发送了一个包含非零返回码的CONNACK报文，它必须将当前会话标志设置为0 [MQTT-3.2.2-4]。

连接返回码 Connect Return code

位置：可变报头的第2个字节。

连接返回码字段使用一个字节的无符号值，在 表格 3.1 –连接返回码的值 中列出。如果服务端收到一个合法的CONNECT报文，但出于某些原因无法处理它，服务端应该尝试发送一个包含非零返回码（表格中的某一个）的CONNACK报文。如果服务端发送了一个包含非零返

回码的CONNACK报文，那么它必须关闭网络连接 [MQTT-3.2.2-5].。

表格 **3.1 –**连接返回码的值

值	返回码响应	描述
0	0x00连接已接受	连接已被服务端接受
1	0x01连接已拒绝，不支持的协议版本	服务端不支持客户端请求的MQTT协议级别
2	0x02连接已拒绝，不合格的客户端标识符	客户端标识符是正确的UTF-8编码，但服务端不允许使用
3	0x03连接已拒绝，服务端不可用	网络连接已建立，但MQTT服务不可用
4	0x04连接已拒绝，无效的用户名或密码	用户名或密码的数据格式无效
5	0x05连接已拒绝，未授权	客户端未被授权连接到此服务器
6-255		保留

如果认为上表中的所有连接返回码都不太合适，那么服务端必须关闭网络连接，不需要发送CONNACK报文 [MQTT-3.2.2-6]。

3.2.3 有效载荷

CONNACK报文没有有效载荷。

第三章目录 MQTT控制报文

- 3.14 DISCONNECT –断开连接

项目主页

- MQTT协议中文版

3.3 PUBLISH – 发布消息

PUBLISH控制报文是指从客户端向服务端或者服务端向客户端传输一个应用消息。

3.3.1 固定报头

图例 3.10 – PUBLISH报文固定报头描述了固定报头的格式

图例 **3.10 – PUBLISH**报文固定报头

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 1	MQTT控制报文类型 (3)				DUP	QoS-H	QoS-	RETAIN
	0	0	1	1	X	X	X	X
byte 2...	剩余长度							

重发标志 DUP

位置：第1个字节，第3位

如果DUP标志被设置为0，表示这是客户端或服务端第一次请求发送这个PUBLISH报文。如果DUP标志被设置为1，表示这可能是一个早前报文请求的重发。

客户端或服务端请求重发一个PUBLISH报文时，必须将DUP标志设置为1 [MQTT-3.3.1.-1].。对于QoS 0的消息，DUP标志必须设置为0 [MQTT-3.3.1-2]。

服务端发送PUBLISH报文给订阅者时，收到（入站）的PUBLISH报文的DUP标志的值不会被传播。发送（出站）的PUBLISH报文与收到（入站）的PUBLISH报文中的DUP标志是独立设置的，它的值必须单独的根据发送（出站）的PUBLISH报文是否是一个重发来确定 [MQTT-3.3.1-3]。

> 非规范评注
>
> 接收者收到一个DUP标志为1的控制报文时，不能假设它看到了一个这个报文之前的一个副本。
>
> 非规范评注
>
> 需要特别指出的是，DUP标志关注的是控制报文本身，与它包含的应用消息无关。当使用QoS 1时，客户端可能会收到一个DUP标志为0的PUBLISH报文，这个报文包含一个它之前收到过的应用消息的副本，但是用的是不同的报文标识符。 2.3.1节提供了有关报文标识符的更多信息。

服务质量等级 QoS

位置：第1个字节，第2-1位。

这个字段表示应用消息分发的服务质量等级保证。服务质量等级在 表格 3.2 -服务质量定义 中列出。

表格 **3.2 -**服务质量定义

QoS值	**Bit 2**	**Bit 1**	描述
0	0	0	最多分发一次
1	0	1	至少分发一次
2	1	0	只分发一次
-	1	1	保留位

PUBLISH报文不能将QoS所有的位设置为1。如果服务端或客户端收到QoS所有位都为1的PUBLISH报文，它必须关闭网络连接 [MQTT-3.3.1-4]。

保留标志 RETAIN

位置：第1个字节，第0位。

如果客户端发给服务端的PUBLISH报文的保留（RETAIN）标志被设置为1，服务端必须存储这个应用消息和它的服务质量等级（QoS），以便它可以被分发给未来的主题名匹配的订阅者 [MQTT-3.3.1-5]。一个新的订阅建立时，对每个匹配的主题名，如果存在最近保留的消息，它必须被发送给这个订阅者 [MQTT-3.3.1-6]。如果服务端收到一条保留（RETAIN）标志为1的QoS 0消息，它必须丢弃之前为那个主题保留的任何消息。它应该将这个新的QoS 0消息当作那个主题的新保留消息，但是任何时候都可以选择丢弃它 — 如果这种情况发生了，那个主题将没有保留消息 [MQTT-3.3.1-7]。有关存储状态的更多信息见 4.1节 。

服务端发送PUBLISH报文给客户端时，如果消息是作为客户端一个新订阅的结果发送，它必须将报文的保留标志设为1 [MQTT-3.3.1-8]。当一个PUBLISH报文发送给客户端是因为匹配一个已建立的订阅时，服务端必须将保留标志设为0，不管它收到的这个消息中保留标志的值是多少 [MQTT-3.3.1-9]。

保留标志为1且有效载荷为零字节的PUBLISH报文会被服务端当作正常消息处理，它会被发送给订阅主题匹配的客户端。此外，同一个主题下任何现存的保留消息必须被移除，因此这个主题之后的任何订阅者都不会收到一个保留消息 [MQTT-3.3.1-10]。当作正常 意思是现存的客户端收到的消息中保留标志未被设置。服务端不能存储零字节的保留消息 [MQTT-3.3.1-11]。

如果客户端发给服务端的PUBLISH报文的保留标志位0，服务端不能存储这个消息也不能移除或替换任何现存的保留消息 [MQTT-3.3.1-12]。

> 非规范评注
>
> 对于发布者不定期发送状态消息这个场景，保留消息很有用。新的订阅者将会收到最近的状态。

剩余长度字段

等于可变报头的长度加上有效载荷的长度。

3.3.2 可变报头

可变报头按顺序包含主题名和报文标识符。

主题名 Topic Name

主题名（Topic Name）用于识别有效载荷数据应该被发布到哪一个信息通道。

主题名必须是PUBLISH报文可变报头的第一个字段。它必须是 1.5.3节定义的UTF-8编码的字符串 [MQTT-3.3.2-1]。

PUBLISH报文中的主题名不能包含通配符 [MQTT-3.3.2-2]。

服务端发送给订阅客户端的PUBLISH报文的主题名必须匹配该订阅的主题过滤器（根据 4.7节定义的匹配过程）[MQTT-3.3.2-3]。

报文标识符 Packet Identifier

只有当QoS等级是1或2时，报文标识符（Packet Identifier）字段才能出现在PUBLISH报文中。2.3.1节提供了有关报文标识符的更多信息。

可变报头非规范示例

图例 3.11 – PUBLISH报文可变报头非规范示例 举例说明了 表格 3.3 - PUBLISH报文非规范示例 中简要描述的PUBLISH报文的可变报头。

表格 3.3 - PUBLISH报文非规范示例

Field	Value
主题名	a/b
报文标识符	10

图例 **3.11 – PUBLISH**报文可变报头非规范示例

	描述	7	6	5	4	3	2	1	0
Topic Name 主题名									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (3)	0	0	0	0	0	0	1	1
byte 3	‘a’ (0x61)	0	1	1	0	0	0	0	1
byte 4	‘/’ (0x2F)	0	0	1	0	1	1	1	1
byte 5	‘b’ (0x62)	0	1	1	0	0	0	1	0
报文标识符									
byte 6	报文标识符 MSB (0)	0	0	0	0	0	0	0	0
byte 7	报文标识符 LSB (10)	0	0	0	0	1	0	1	0

示例中的主题名为 “a/b”，长度等于3，报文标识符为 “10”

3.3.3 有效载荷

有效载荷包含将被发布的应用消息。数据的内容和格式是应用特定的。有效载荷的长度这样计算：用固定报头中的剩余长度字段的值减去可变报头的长度。包含零长度有效载荷的PUBLISH报文是合法的。

3.3.4 响应

PUBLISH报文的接收者必须按照根据PUBLISH报文中的QoS等级发送响应，见下面表格的描述 [MQTT-3.3.4-1]。

表格 **3.4 – PUBLISH**报文的预期响应

服务质量等级	预期响应
QoS 0	无响应
QoS 1	PUBACK报文
QoS 2	PUBREC报文

3.3.5 动作 Actions

客户端使用PUBLISH报文发送应用消息给服务端，目的是分发到其它订阅匹配的客户端。

服务端使用PUBLISH报文发送应用消息给每一个订阅匹配的客户端。

客户端使用带通配符的主题过滤器请求订阅时，客户端的订阅可能会重复，因此发布的消息可能会匹配多个过滤器。对于这种情况，服务端必须将消息分发给所有订阅匹配的QoS等级最高的客户端 [MQTT-3.3.5-1]。服务端之后可以按照订阅的QoS等级，分发消息的副本给每一个匹配的订阅者。

收到一个PUBLISH报文时，接收者的动作取决于4.3节描述的QoS等级。

如果服务端实现不授权某个客户端发布PUBLISH报文，它没有办法通知那个客户端。它必须按照正常的QoS规则发送一个正面的确认，或者关闭网络连接 [MQTT-3.3.5-2]。

第三章目录 MQTT控制报文

项目主页

- MQTT协议中文版

3.4 PUBACK –发布确认

PUBACK报文是对QoS 1等级的PUBLISH报文的响应。

3.4.1 固定报头

图例 **3.12 - PUBACK**报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT报文类型 (4)				保留位			
	0	1	0	0	0	0	0	0
byte 2...	剩余长度							
	0	0	0	0	0	0	1	0

剩余长度字段

表示可变报头的长度。对PUBACK报文这个值等于2.

3.4.2 可变报头

包含等待确认的PUBLISH报文的报文标识符。

图例 **3.13 – PUBACK**报文可变报头

Bit	7	6	5	4	3	2	1	0
byte 1	报文标识符 MSB							
byte 2	报文标识符 LSB							

3.4.3 有效载荷

PUBACK报文没有有效载荷。

3.4.4 动作

完整的描述见4.3.2节。

第三章目录 MQTT控制报文

项目主页

- MQTT协议中文版

3.5 PUBREC – 发布收到（QoS 2，第一步）

PUBREC报文是对QoS等级2的PUBLISH报文的响应。它是QoS 2等级协议交换的第二个报文。

3.5.1 固定报头

图例 **3.14 – PUBREC**报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT控制报文类型 (5)				保留位			
	0	1	0	1	0	0	0	0
byte 2	剩余长度 (2)							
	0	0	0	0	0	0	1	0

剩余长度字段

表示可变报头的长度。对PUBREC报文它的值等于2。

3.5.2 可变报头

可变报头包含等待确认的PUBLISH报文的报文标识符。

图例 **3.15 – PUBREC**报文可变报头

Bit	7	6	5	4	3	2	1	0
byte 1	报文标识符 MSB							
byte 2	报文标识符 LSB							

3.5.3 有效载荷

PUBREC报文没有有效载荷。

3.5.4 动作

完整的描述见 4.3.3节。

第三章目录 **MQTT**控制报文

项目主页

- MQTT协议中文版

3.6 PUBREL – 发布释放（QoS 2，第二步）

PUBREL报文是对PUBREC报文的响应。它是QoS 2等级协议交换的第三个报文。

3.6.1 固定报头

图例 **3.16 – PUBREL**报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT控制报文类型 (6)				保留位			
	1	1	0	0	0	0	1	0
byte 2	剩余长度 (2)							
	0	0	0	0	0	0	1	0

PUBREL控制报文固定报头的第3,2,1,0位是保留位，必须被设置为0,0,1,0。服务端必须将其它的任何值都当做是不合法的并关闭网络连接 [MQTT-3.6.1-1]。

剩余长度字段

表示可变报头的长度。对PUBREL报文这个值等于2.

3.6.2 可变报头

可变报头包含与等待确认的PUBREC报文相同的报文标识符。

图例 **3.17 – PUBREL**报文可变报头

Bit	7	6	5	4	3	2	1	0
byte 1	报文标识符 MSB							
byte 2	报文标识符 LSB							

3.6.3 有效载荷

PUBREL报文没有有效载荷。

3.6.4 动作

完整的描述见 4.3.3节。

第三章目录 MQTT控制报文

项目主页

- MQTT协议中文版

3.7 PUBCOMP – 发布完成（QoS 2，第三步）

PUBCOMP报文是对PUBREL报文的响应。它是QoS 2等级协议交换的第四个也是最后一个报文。

3.7.1 固定报头

图例 **3.18 – PUBCOMP**报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT控制报文类型 (7)				保留位			
	0	1	1	1	0	0	0	0
byte 2	剩余长度 (2)							
	0	0	0	0	0	0	1	0

剩余长度字段

表示可变报头的长度。对PUBCOMP报文这个值等于2。

3.7.2 可变报头

可变报头包含与等待确认的PUBREL报文相同的报文标识符。

图例 **3.19 – PUBCOMP**报文可变报头

Bit	7	6	5	4	3	2	1	0
byte 1	报文标识符 MSB							
byte 2	报文标识符 LSB							

3.7.3 有效载荷

PUBCOMP报文没有有效载荷。

3.7.4 动作

完整的描述见4.3.3节。

第三章目录 **MQTT**控制报文

项目主页

- MQTT协议中文版

3.8 SUBSCRIBE - 订阅主题

客户端向服务端发送SUBSCRIBE报文用于创建一个或多个订阅。每个订阅注册客户端关心的一个或多个主题。为了将应用消息转发给与那些订阅匹配的主题，服务端发送PUBLISH报文给客户端。SUBSCRIBE报文也（为每个订阅）指定了最大的QoS等级，服务端根据这个发送应用消息给客户端。

3.8.1 固定报头

图例 **3.20 – SUBSCRIBE**报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT控制报文类型 (8)	保留位						
	1	0	0	0	0	0	1	0
byte 2	剩余长度							

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT控制报文类型 (8)				保留位			
	1	0	0	0	0	0	0	0
byte 2	剩余长度							

SUBSCRIBE控制报固定报头的第3,2,1,0位是保留位，必须分别设置为0,0,1,0。服务端必须将其它的任何值都当做是不合法的并关闭网络连接 [MQTT-3.8.1-1]。

剩余长度字段

等于可变报头的长度（2字节）加上有效载荷的长度。

3.8.2可变报头

可变报头包含报文标识符。2.3.1提供了有关报文标识符的更多信息。

可变报头非规范示例

图例 3.21 – 报文标识符等于10的可变报头，非规范示例 展示了报文标识符设置为10时的可变报头。

图例 **3.21 –** 报文标识符等于**10**的可变报头，非规范示例

	描述	7	6	5	4	3	2	1	0
报文标识符									
byte 1	报文标识符 MSB (0)	0	0	0	0	0	0	0	0
byte 2	报文标识符 LSB (10)	0	0	0	0	1	0	1	0

3.8.3 有效载荷

SUBSCRIBE报文的有效载荷包含了一个主题过滤器列表，它们表示客户端想要订阅的主题。SUBSCRIBE报文有效载荷中的主题过滤器列表必须是1.5.3节定义的UTF-8字符串 [MQTT-3.8.3-1]。服务端应该支持包含通配符（4.7.1节定义的）的主题过滤器。如果服务端选择不支持包含通配符的主题过滤器，必须拒绝任何包含通配符过滤器的订阅请求 [MQTT-3.8.3-2]。每一个过滤器后面跟着一个字节，这个字节被叫做 服务质量要求（Requested QoS）。它给出了服务端向客户端发送应用消息所允许的最大QoS等级。

SUBSCRIBE报文的有效载荷必须包含至少一对主题过滤器 和 QoS等级字段组合。没有有效载荷的SUBSCRIBE报文是违反协议的 [MQTT-3.8.3-3]。有关错误处理的信息请查看4.8节。

请求的最大服务质量等级字段编码为一个字节，它后面跟着UTF-8编码的主题名，那些主题过滤器 /和QoS等级组合是连续地打包。

图例 **3.22 – SUBSCRIBE**报文有效载荷格式

描述	7	6	5	4	3	2	1	0
主题过滤器								
byte 1	长度 MSB							
byte 2	长度 LSB							
bytes 3..N	主题过滤器（Topic Filter）							
服务质量要求（Requested QoS）								
	保留位	服务质量等级						
byte N+1	0	0	0	0	0	0	X	X

描述	7	6	5	4	3	2	1	0
主题过滤器								
byte 1	长度 MSB							
byte 2	长度 LSB							
byte 3..N	主题过滤器（Topic Filter）							
服务质量要求（Requested QoS）								
	保留位						服务质量等级	
byte N+1	0	0	0	0	0	0	X	X

当前版本的协议没有用到服务质量要求（Requested QoS）字节的高六位。如果有效载荷中的任何位是非零值，或者QoS不等于0,1或2，服务端必须认为SUBSCRIBE报文是不合法的并关闭网络连接 [MQTT-3-8.3-4]。

有效载荷非规范示例

> 图例 3.23 – 有效载荷字节格式非规范示例 展示了 表格 3.5 – 有效载荷非规范示例 中简略描述的SUBSCRIBE报文的有效载荷。

表格 **3.5 –** 有效载荷非规范示例

主题名	**"a/b"**
服务质量要求	0x01
主题名	"c/d"
服务质量要求	0x02

图例 **3.23 –** 有效载荷字节格式非规范示例

	描述	7	6	5	4	3	2	1	0
主题过滤器（Topic Filter）									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (3)	0	0	0	0	0	0	1	1
byte 3	‘a’ (0x61)	0	1	1	0	0	0	0	1
byte 4	‘/’ (0x2F)	0	0	1	0	1	1	1	1
byte 5	‘b’ (0x62)	0	1	1	0	0	0	1	0
服务质量要求（Requested QoS）									
byte 6	Requested QoS(1)	0	0	0	0	0	0	0	1
主题过滤器（Topic Filter）									
byte 7	Length MSB (0)	0	0	0	0	0	0	0	0
byte 8	Length LSB (3)	0	0	0	0	0	0	1	1
byte 9	‘c’ (0x63)	0	1	1	0	0	0	1	1
byte 10	‘/’ (0x2F)	0	0	1	0	1	1	1	1
byte 11	‘d’ (0x64)	0	1	1	0	0	1	0	0
服务质量要求（Requested QoS）									
byte 12	Requested QoS(2)	0	0	0	0	0	0	1	0

3.8.4 响应

服务端收到客户端发送的一个SUBSCRIBE报文时，必须使用SUBACK报文响应 [MQTT-3.8.4-1]。SUBACK报文必须和等待确认的SUBSCRIBE报文有相同的报文标识符 [MQTT-3.8.4-2]。

允许服务端在发送SUBACK报文之前就开始发送与订阅匹配的PUBLISH报文。

如果服务端收到一个SUBSCRIBE报文，报文的主题过滤器与一个现存订阅的主题过滤器相同，那么必须使用新的订阅彻底替换现存的订阅。新订阅的主题过滤器和之前订阅的相同，但是它的最大QoS值可以不同。与这个主题过滤器匹配的任何现存的保留消息必须被重发，但是发布流程不能中断 [MQTT-3.8.4-3]。

如果主题过滤器不同于任何现存订阅的过滤器，服务端会创建一个新的订阅并发送所有匹配的保留消息。

如果服务端收到包含多个主题过滤器的SUBSCRIBE报文，它必须如同收到了一系列的多个SUBSCRIBE报文一样处理那个，除了需要将它们的响应合并到一个单独的SUBACK报文发送 [MQTT-3.8.4-4]。

服务端发送给客户端的SUBACK报文对每一对主题过滤器 和QoS等级都必须包含一个返回码。这个返回码必须表示那个订阅被授予的最大QoS等级，或者表示这个订阅失败 [MQTT-3.8.4-5]。服务端可以授予比订阅者要求的低一些的QoS等级。为响应订阅而发出的消息的有效载荷的QoS必须是原始发布消息的QoS和服务端授予的QoS两者中的最小值。如果原始消息的QoS是1而被授予的最大QoS是0，允许服务端重复发送一个消息的副本给订阅者 [MQTT-3.8.4-6]。

> 非规范示例 对某个特定的主题过滤器，如果正在订阅的客户端被授予的最大QoS等级是1，那么匹配这个过滤器的QoS等级0的应用消息会按QoS等级0分发给这个客户端。这意味着客户端最多收到这个消息的一个副本。从另一方面说，发布给同一主题的QoS等级2的消息会被服务端降级到QoS等级1再分发给客户端，因此客户端可能会收到重复的消息副本。
>
> 如果正在订阅的客户端被授予的最大QoS等级是0，那么原来按QoS等级2发布给客户端的应用消息在繁忙时可能会丢失，但是服务端不应该发送重复的消息副本。发布给同一主题的 QoS等级1的消息在传输给客户端时可能会丢失或重复。
>
> 非规范评注
>
> 使用QoS等级2订阅一个主题过滤器等于是说：我想要按照它们发布时的*QoS*等级接受匹配这个过滤器的消息 。这意味着，确定消息分发时可能的最大QoS等级是发布者的责任，而订阅者可以要求服务端降低QoS到更适合它的等级。

第三章目录 MQTT控制报文

- 3.14 DISCONNECT –断开连接

项目主页

- MQTT协议中文版

3.9 SUBACK – 订阅确认

服务端发送SUBACK报文给客户端，用于确认它已收到并且正在处理SUBSCRIBE报文。

SUBACK报文包含一个返回码清单，它们指定了SUBSCRIBE请求的每个订阅被授予的最大QoS等级。

3.9.1 固定报头

图例 **3.24 – SUBACK**报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT控制报文类型 (9)				保留位			
	1	0	0	1	0	0	0	0
byte 2	剩余长度							

剩余长度字段

等于可变报头的长度加上有效载荷的长度。

3.9.2 可变报头

可变报头包含等待确认的SUBSCRIBE报文的报文标识符。图例 3.25 – SUBACK报文可变报头 描述了可变报头的格式。

图例 **3.25 – SUBACK**报文可变报头

Bit	7	6	5	4	3	2	1	0
byte 1	报文标识符 MSB							
byte 2	报文标识符 LSB							

3.9.3 有效载荷

有效载荷包含一个返回码清单。每个返回码对应等待确认的SUBSCRIBE报文中的一个主题过滤器。返回码的顺序必须和SUBSCRIBE报文中主题过滤器的顺序相同 [MQTT-3.9.3-1]。

图例 3.26 – SUBACK报文有效载荷格式 描述了有效载荷中单字节编码的返回码字段。

图例 **3.26 – SUBACK**报文有效载荷格式

Bit	7	6	5	4	3	2	1	0
	返回码							
byte 1	X	0	0	0	0	0	X	X

允许的返回码值：

- 0x00 - 最大QoS 0
- 0x01 - 成功 – 最大QoS 1
- 0x02 - 成功 – 最大 QoS 2
- 0x80 - Failure 失败

0x00, 0x01, 0x02, 0x80之外的SUBACK返回码是保留的，不能使用[MQTT-3.9.3-2]。

有效载荷非规范示例

> 图例 3.27 -有效载荷字节格式非规范示例 展示了在 表格 3.6 -有效载荷非规范示例 简要描述的SUBACK报文的有效载荷。

表格 **3.6 -**有效载荷非规范示例

Success - Maximum QoS 0	0
Success - Maximum QoS 2	2
Failure	128

图例 **3.27 -**有效载荷字节格式非规范示例

	描述	7	6	5	4	3	2	1	0
byte 1	Success - Maximum QoS 0	0	0	0	0	0	0	0	0
byte 2	Success - Maximum QoS 2	0	0	0	0	0	0	1	0
byte 3	Failure	1	0	0	0	0	0	0	0

第三章目录 MQTT控制报文

项目主页

- MQTT协议中文版

3.10 UNSUBSCRIBE –取消订阅

客户端发送UNSUBSCRIBE报文给服务端，用于取消订阅主题。

3.10.1 固定报头

图例 **3.28 – UNSUBSCRIBE**报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT控制报文类型 (10)				保留位			
	1	0	1	0	0	0	1	0
byte 2	剩余长度							

UNSUBSCRIBE报文固定报头的第3,2,1,0位是保留位且必须分别设置为0,0,1,0。服务端必须认为任何其它的值都是不合法的并关闭网络连接 [MQTT-3.10.1-1]。

剩余长度字段

等于可变报头的长度加上有效载荷的长度。

3.10.2 可变报头

可变报头包含一个报文标识符。2.3.1节提供了有关报文标识符的更多信息。

图例 **3.29 – UNSUBSCRIBE**报文可变报头

Bit	7	6	5	4	3	2	1	0
byte 1	报文标识符 MSB							
byte 2	报文标识符 LSB							

3.10.3 有效载荷

UNSUBSCRIBE报文的有效载荷包含客户端想要取消订阅的主题过滤器列表。UNSUBSCRIBE报文中的主题过滤器必须是连续打包的、按照1.5.3节定义的UTF-8编码字符串 [MQTT-3.10.3-1]。

UNSUBSCRIBE报文的有效载荷必须至少包含一个消息过滤器。没有有效载荷的UNSUBSCRIBE报文是违反协议的 [MQTT-3.10.3-2]。有关错误处理的更多信息请查看4.8节。

有效载荷非规范示例

> 图例 3.30 -有效载荷字节格式非规范示例 展示了 表格 3.7 -有效载荷非规范示例 简要描述的UNSUBSCRIBE报文的有效载荷。

表格 **3.7** -有效载荷非规范示例

主题过滤器	**“a/b”**
主题过滤器	“c/d”

图例 **3.30** -有效载荷字节格式非规范示例

	描述	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
主题过滤器									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (3)	0	0	0	0	0	0	1	1
byte 3	‘a’ (0x61)	0	1	1	0	0	0	0	1
byte 4	‘/’ (0x2F)	0	0	1	0	1	1	1	1
byte 5	‘b’ (0x62)	0	1	1	0	0	0	1	0
主题过滤器									
byte 6	Length MSB (0)	0	0	0	0	0	0	0	0
byte 7	Length LSB (3)	0	0	0	0	0	0	1	1
byte 8	‘c’ (0x63)	0	1	1	0	0	0	1	1
byte 9	‘/’ (0x2F)	0	0	1	0	1	1	1	1
byte 10	‘d’ (0x64)	0	1	1	0	0	1	0	0

3.10.4 响应

UNSUBSCRIBE报文提供的主题过滤器（无论是否包含通配符）必须与服务端持有的这个客户端的当前主题过滤器集合逐个字符比较。如果有任何过滤器完全匹配，那么它（服务端）自己的订阅将被删除，否则不会有进一步的处理 [MQTT-3.10.4-1]。

如果服务端删除了一个订阅：

- 它必须停止分发任何新消息给这个客户端 [MQTT-3.10.4-2]。
- 它必须完成分发任何已经开始往客户端发送的QoS 1和QoS 2的消息 [MQTT-3.10.4-3]。
- 它可以继续发送任何现存的准备分发给客户端的缓存消息。

服务端必须发送UNSUBACK报文响应客户端的UNSUBSCRIBE请求。UNSUBACK报文必须包含和UNSUBSCRIBE报文相同的报文标识符 [MQTT-3.10.4-4]。即使没有删除任何主题订阅，服务端也必须发送一个UNSUBACK响应 [MQTT-3.10.4-5]。

如果服务端收到包含多个主题过滤器的UNSUBSCRIBE报文，它必须如同收到了一系列的多个UNSUBSCRIBE报文一样处理那个报文，除了将它们的响应合并到一个单独的UNSUBACK报文外。 [MQTT-3.10.4-6]。

第三章目录 MQTT控制报文

项目主页

- MQTT协议中文版

3.11 UNSUBACK – 取消订阅确认

服务端发送UNSUBACK报文给客户端用于确认收到UNSUBSCRIBE报文。

3.11.1 固定报头

图例 **3.31 – UNSUBACK**报文固定报头

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 1	MQTT控制报文类型 (11)	保留位						
	1	0	1	1	0	0	0	0
byte 2	剩余长度 (2)							
	0	0	0	0	0	0	1	0

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 1	MQTT控制报文类型 (11)				保留位			
	1	0	1	1	0	0	0	0
byte 2	剩余长度 (2)							
	0	0	0	0	0	0	1	0

剩余长度字段

表示可变报头的长度，对UNSUBACK报文这个值等于2。

3.11.2 可变报头

可变报头包含等待确认的UNSUBSCRIBE报文的报文标识符。

图例 **3.32 – UNSUBACK**报文可变报头

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 1	报文标识符 MSB							
byte 2	报文标识符 LSB							

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 1	报文标识符 MSB							
byte 2	报文标识符 LSB							

3.11.3 有效载荷

UNSUBACK报文没有有效载荷。

第三章目录 MQTT控制报文

项目主页

- MQTT协议中文版

3.12 PINGREQ – 心跳请求

客户端发送PINGREQ报文给服务端的。用于：

1. 在没有任何其它控制报文从客户端发给服务的时，告知服务端客户端还活着。
2. 请求服务端发送 响应确认它还活着。
3. 使用网络以确认网络连接没有断开。

保持连接（Keep Alive）处理中用到这个报文，详细信息请查看 3.1.2.10节。

3.12.1 固定报头

图例 **3.33 – PINGREQ**报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT控制报文类型 (12)	保留位						
	1	1	0	0	0	0	0	0
byte 2	剩余长度 (0)							
	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT控制报文类型 (12)				保留位			
	1	1	0	0	0	0	0	0
byte 2	剩余长度 (0)							
	0	0	0	0	0	0	0	0

3.12.2 可变报头

PINGREQ报文没有可变报头。

3.12.3 有效载荷

PINGREQ报文没有有效载荷。

3.12.4 响应

服务端必须发送 PINGRESP报文响应客户端的PINGREQ报文 [MQTT-3.12.4-1]。

第三章目录 MQTT控制报文

项目主页

- MQTT协议中文版

3.13 PINGRESP – 心跳响应

服务端发送PINGRESP报文响应客户端的PINGREQ报文。表示服务端还活着。

保持连接（Keep Alive）处理中用到这个报文，详情请查看 3.1.2.10节。

3.13.1 固定报头

图例 **3.34 – PINGRESP**报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT控制报文类型 (13)				保留位			
	1	1	0	1	0	0	0	0
byte 2	剩余长度 (0)							
	0	0	0	0	0	0	0	0

3.13.2 可变报头

PINGRESP报文没有可变报头。

3.13.3 有效载荷

PINGRESP报文没有有效载荷。

第三章目录 **MQTT**控制报文

- 3.12 PINGREQ – 心跳请求
- 3.13 PINGRESP – 心跳响应
- 3.14 DISCONNECT –断开连接

项目主页

- MQTT协议中文版

3.14 DISCONNECT –断开连接

DISCONNECT报文是客户端发给服务端的最后一个控制报文。表示客户端正常断开连接。

3.14.1 固定报头

图例 **3.35 – DISCONNECT**报文固定报头

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT控制报文类型 (14)				保留位			
	1	1	1	0	0	0	0	0
byte 2	剩余长度 (0)							
	0	0	0	0	0	0	0	0

服务端必须验证所有的保留位都被设置为0，如果它们不为0必须断开连接 [MQTT-3.14.1-1]。

3.14.2 可变报头

DISCONNECT报文没有可变报头。

3.14.3 有效载荷

DISCONNECT报文没有有效载荷。

3.14.4 响应

客户端发送DISCONNECT报文之后：

- 必须关闭网络连接 [MQTT-3.14.4-1]。
- 不能通过那个网络连接再发送任何控制报文 [MQTT-3.14.4-2]。

服务端在收到DISCONNECT报文时：

- 必须丢弃任何与当前连接关联的未发布的遗嘱消息，具体描述见 3.1.2.5节 [MQTT-3.14.4-3]。
- 应该关闭网络连接，如果客户端 还没有这么做。

第三章目录 MQTT控制报文

第三章目录 MQTT控制报文

- 3.0 Contents – MQTT控制报文
- 3.1 CONNECT – 连接服务端
- 3.2 CONNACK – 确认连接请求
- 3.3 PUBLISH – 发布消息
- 3.4 PUBACK –发布确认
- 3.5 PUBREC – 发布收到（QoS 2，第一步）
- 3.6 PUBREL – 发布释放（QoS 2，第二步）
- 3.7 PUBCOMP – 发布完成（QoS 2，第三步）
- 3.8 SUBSCRIBE - 订阅主题
- 3.9 SUBACK – 订阅确认
- 3.10 UNSUBSCRIBE –取消订阅
- 3.11 UNSUBACK – 取消订阅确认
- 3.12 PINGREQ – 心跳请求
- 3.13 PINGRESP – 心跳响应
- 3.14 DISCONNECT –断开连接

项目主页

- MQTT协议中文版

第四章 操作行为 Operational behavior

目录

4.1 状态存储 Storing state

为了提供服务质量保证，客户端和服务端有必要存储会话状态。在整个会话期间，客户端和服务端都必须存储会话状态 [MQTT-4.1.0-1]。会话必须至少持续和它的活跃网络连接同样长的时间 [MQTT-4.1.0-2]。

服务端的保留消息不是会话状态的组成部分。服务端应该保留那种消息直到客户端删除它。

> 非规范评注
>
> 客户端和服务端实现的存储容量必然是有限的，还可能要受管理策略的限制，比如跨网络连接的会话状态的最大存储时间。已保存的会话状态丢失可能是某个管理操作造成的，例如对某个预定义条件的自动响应。它造成的后果就是会话终止。这些操作可能是资源限制或其他操作原因引发的。需要谨慎的评估客户端和服务端的存储容量，以确保存储空间够用。
>
> 非规范评注
>
> 客户端或服务端的软硬件故障都可能导致会话状态的丢失或损坏。
>
> 非规范评注
>
> 服务器和客户端操作正常可能意味着，已保存的状态丢失或损坏是管理操作或软硬件故障造成的。管理操作可能是对某个预定义条件的自动响应。这些操作可能是资源限制或其他操作原因引发的。例如，服务端可能会基于外部条件，决定不再将某个消息或某些消息分发给任何当前的或以后的客户端。
>
> 非规范评注
>
> MQTT用户应该评估MQTT客户端和服务端实现的存储容量，确保能满足需求。

非规范示例

例如，想要收集电表读数的用户可能会决定使用QoS 1等级的消息，因为他们不能接受数据在网络传输途中丢失，但是，他们可能认为客户端和服务端的数据可以存储在内存（易失性存储器）中，因为（他们觉得）电力供应是非常可靠的，不会有太大的数据丢失风险。

与之相反，停车计费支付应用的提供商可能决定任何情况下都不能让数据支付消息丢失，因此他们要求在通过网络传输之前，所有的数据必须写入到非易失性存储器中（如硬盘）。

4.2 网络连接 Network Connections

MQTT协议要求基础传输层能够提供有序的、可靠的、双向传输（从客户端到服务端 和从服务端到客户端）的字节流。

> 非规范评注
>
> MQTT 3.1使用的传输层协议是 [RFC793] 定义的TCP/IP协议。下面的协议也支持：

- TLS协议 [RFC5246]
- WebSocket协议 [RFC6455]

> 非规范评注
>
> TCP端口8883和1883已在IANA注册，分别用于MQTT的TLS和非TLS通信。

无连接的网络传输协议如UDP是不支持的，因为他们可能会丢失数据包或对数据包重排序。

4.3 服务质量等级和协议流程 QoS

MQTT按照这里定义的服务质量 (QoS) 等级分发应用消息。分发协议是对称的，在下面的描述中，客户端和服务端既可以是发送者也可以是接收者。分发协议关注的是从单个发送者到单个接收者的应用消息。服务端分发应用消息给多个客户端时，每个客户端独立处理。分发给客户端的出站应用消息和入站应用消息的QoS等级可能是不同的。

下面的非规范流程图展示了可能的实现方法。

4.3.1 QoS 0:最多分发一次

消息的分发依赖于底层网络的能力。接收者不会发送响应，发送者也不会重试。消息可能送达一次也可能根本没送达。

对于QoS 0的分发协议，发送者

- 必须发送QoS等于0，DUP等于0的PUBLISH报文 [MQTT-4.3.1-1]。

对于QoS 0的分发协议，接收者

- 接受PUBLISH报文时同时接受消息的所有权。

图例 **4.1 – QoS 0**协议流程图，非规范示例

发送者动作	控制报文	接收者动作
PUBLISH 报文 QoS 0, DUP=0		
	---------->	
		分发应用消息给适当的后续接收者（们）

4.3.2 QoS 1: 至少分发一次

服务质量确保消息至少送达一次。QoS 1的PUBLISH报文的可变报头中包含一个报文标识符，需要PUBACK报文确认。2.3.1节提供了有关报文标识符的更多信息。

对于QoS 1的分发协议，发送者

- 每次发送新的应用消息都必须分配一个未使用的报文标识符。
- 发送的PUBLISH报文必须包含报文标识符且QoS等于1，DUP等于0。

- 必须将这个PUBLISH报文看作是 未确认的 ，直到从接收者那收到对应的PUBACK报文。4.4节有一个关于未确认消息的讨论。

> [MQTT-4.3.2-1].
>
> 一旦发送者收到PUBACK报文，这个报文标识符就可以重用。

注意：允许发送者在等待确认时使用不同的报文标识符发送后续的PUBLISH报文。

对于QoS 1的分发协议，接收者

- 响应的PUBACK报文必须包含一个报文标识符，这个标识符来自接收到的、已经接受所有权的PUBLISH报文。
- 发送了PUBACK报文之后，接收者必须将任何包含相同报文标识符的入站PUBLISH报文当作一个新的消息，并忽略它的DUP标志的值。

> [MQTT-4.3.2-2].

图例 **4.2 – QoS 1**协议流程图，非规范示例

发送者动作	控制报文	接收者动作
存储消息		
发送PUBLISH报文 QoS=1, DUP=0，带报文标识符	--------->	
		开始应用消息的后续分发[1]
	<----------	发送PUBACK报文，带报文标识符
丢弃消息		

> [1]不要求接收者在发送PUBACK之前完整分发应用消息。原来的发送者收到PUBACK报文之后，应用消息的所有权就会转移给这个接收者。

4.3.3 QoS 2: 仅分发一次

这是最高等级的服务质量，消息丢失和重复都是不可接受的。使用这个服务质量等级会有额外的开销。

QoS 2的消息可变报头中有报文标识符。2.3.1节提供了有关报文标识符的更多信息。QoS 2的PUBLISH报文的接收者使用一个两步确认过程来确认收到。

对于QoS 2的分发协议，发送者

- 必须给要发送的新应用消息分配一个未使用的报文标识符。
- 发送的PUBLISH报文必须包含报文标识符且报文的QoS等于2,，DUP等于0。

- 必须将这个PUBLISH报文看作是 未确认的 ，直到从接收者那收到对应的PUBREC报文。4.4节有一个关于未确认消息的讨论。
- 收到PUBREC报文后必须发送一个PUBREL报文。PUBREL报文必须包含与原始PUBLISH报文相同的报文标识符。
- 必须将这个PUBREL报文看作是 未确认的 ，直到从接收者那收到对应的PUBCOMP报文。
- 一旦发送了对应的PUBREL报文就不能重发这个PUBLISH报文。

[MQTT-4.3.3-1].

> 一旦发送者收到PUBCOMP报文，这个报文标识符就可以重用。

注意：允许发送者在等待确认时使用不同的报文标识符发送后续的PUBLISH报文。

对于QoS 2的分发协议，接收者

- 响应的PUBREC报文必须包含报文标识符，这个标识符来自接收到的、已经接受所有权的PUBLISH报文。
- 在收到对应的PUBREL报文之前，接收者必须发送PUBREC报文确认任何后续的具有相同标识符的PUBLISH报文。 在这种情况下，它不能重复分发消息给任何后续的接收者。
- 响应PUBREL报文的PUBCOMP报文必须包含与PUBREL报文相同的标识符。
- 发送PUBCOMP报文之后，接收者必须将包含相同报文标识符的任何后续PUBLISH报文当作一个新的发布。

[MQTT-4.3.3-2].

图例 **4.3 – QoS 2**协议流程图，非规范示例

发送者动作	控制报文	接收者动作
存储消息		
发送PUBLISH报文，QoS=2, DUP=0，带报文标识符		
	---------->	
		方法A：存储消息，或方法B：存储报文标识符，然后开始向前分发这个应用消息[1]。
		发送PUBREC报文，带报文标识符。
	<----------	
丢弃消息，存储PUBREC中的报文标识符		
发送PUBREL报文，带报文标识符		
	---------->	
		方法A：开始向前分发应用消息[1]然后丢弃消息或方法B：丢弃报文标识符
		发送PUBCOMP报文，带报文标识符
	<----------	
丢弃已保存的状态		

> [1] 不要求接收者在发送PUBREC或PUBCOMP之前完整分发应用消息。原来的发送者收到PUBREC报文之后，应用消息的所有权就会转移给这个接收者。
>
> 图例 4.3 – QoS 2协议流程图，非规范示例 展示了接收者对QoS 2等级消息的两种处理方法。他们的区别是消息什么时候可以开始分发。实现者可以决定使用哪种方法。只要实现者只选择了一种方法，就不会影响QoS流程的可靠性。

4.4 消息分发重试 Message delivery retry

客户端设置清理会话（CleanSession）标志为0重连时，客户端和服务端必须使用原始的报文标识符重发任何未确认的PUBLISH报文（如果QoS>0）和PUBREL报文 [MQTT-4.4.0-1]。这是唯一要求客户端或服务端重发消息的情况。

> 非规范评注
>
> 控制报文的重发曾经需要克服某些陈旧TCP网络上的数据丢失问题。部署在那些环境中的MQTT 3.1.1实现可能仍然需要关注这个问题。

4.5 消息收到 Message receipt

服务端接管入站应用消息的所有权时，它必须将消息添加到订阅匹配的客户端的会话状态中。匹配规则定义见 4.7节 [MQTT-4.5.0-1]。

正常情况下，客户端收到发送给它的订阅的消息。客户端也可能收到不是与它的订阅精确匹配的消息。如果服务端自动给客户端分配了一个订阅，可能发生这种情况。正在处理UBSUBSCRIBE请求时也可能收到消息。客户端必须按照可用的服务质量（QoS）规则确认它收到的任何PUBLISH报文，不管它选择是否处理报文包含的应用消息 [MQTT-4.5.0-2]。

4.6 消息排序 Message ordering

实现本章定义的协议流程时，客户端必须遵循下列规则：

- 重发任何之前的PUBLISH报文时，必须按原始PUBLISH报文的发送顺序重发（适用于QoS 1和QoS 2消息）[MQTT-4.6.0-1]。
- 必须按照对应的PUBLISH报文的顺序发送PUBACK报文（QoS 1消息）[MQTT-4.6.0-2]。
- 必须按照对应的PUBLISH报文的顺序发送PUBREC报文（QoS 2消息）[MQTT-4.6.0-3]。
- 必须按照对应的PUBREC报文的顺序发送PUBREL报文（QoS 2消息）[MQTT-4.6.0-4]。

服务端必须默认认为每个主题都是有序的。它可以提供一个管理功能或其它机制，以允许将一个或多个主题当作是无序的 [MQTT-4.6.0-5]。

服务端处理发送给有序主题的消息时，必须按照上面的规则将消息分发给每个订阅者。此外，它必须按照从客户端收到的顺序发送PUBLISH报文给消费者（对相同的主题和QoS）[MQTT-4.6.0-6]。

> 非规范评注
>
> 上面列出的规则确保，使用QoS 1发布和订阅的消息流，订阅者按照消息发布时的顺序收到每条消息的最终副本，但是消息可能会重复，这可能导致在它的后继消息之后收到某个已经收到消息的重发版本。例如，发布者按顺序1,2,3,4发送消息，订阅者收到的顺序可能是1,2,3,2,3,4。
>
> 如果客户端和服务端能保证任何时刻最多有一条消息在 传输中（*in-flight*）（在某条消息被确认前不发送后面的那条消息），那么，不会有QoS 1的消息会在它的任何后续消息之后收到。 例如，订阅者收到的顺序可能是1,2,3,3,4，而不是1,2,3,2,3,4。将传输窗口（in-flight window）设为1意味着，在同一个主题上，即使发布者发送了一系列不同QoS等级的消息，它们的顺序也被保留。

4.7 主题名和主题过滤器 Topic Names and Topic Filters

4.7.1 主题通配符 Topic wildcards

主题层级（topic level）分隔符用于将结构化引入主题名。如果存在分隔符，它将主题名分割为多个主题层级 *topic level* 。

订阅的主题过滤器可以包含特殊的通配符，允许你一次订阅多个主题。

主题过滤器中可以使用通配符，但是主题名不能使用通配符 [MQTT-4.7.1-1]。

主题层级分隔符 Topic level separator

斜杠（‘/’ U+002F）用于分割主题的每个层级，为主题名提供一个分层结构。当客户端订阅指定的主题过滤器包含两种通配符时，主题层级分隔符就很有用了。主题层级分隔符可以出现在主题过滤器或主题名字的任何位置。相邻的主题层次分隔符表示一个零长度的主题层级。

多层通配符 Multi-level wildcard

数字标志（‘#’ U+0023）是用于匹配主题中任意层级的通配符。多层通配符表示它的父级和任意数量的子层级。多层通配符必须位于它自己的层级或者跟在主题层级分隔符后面。不管哪种情况，它都必须是主题过滤器的最后一个字符 [MQTT-4.7.1-2]。

> 非规范评注
>
> 例如，如果客户端订阅主题 “sport/tennis/player1/#”，它会收到使用下列主题名发布的消息：

- “sport/tennis/player1”

- “sport/tennis/player1/ranking”
- “sport/tennis/player1/score/wimbledon”

> 非规范评注

- “sport/#”也匹配单独的 “sport” ，因为 # 包括它的父级。
- “#”是有效的，会收到所有的应用消息。
- “sport/tennis/#”也是有效的。
- “sport/tennis#”是无效的。
- “sport/tennis/#/ranking”是无效的。

单层通配符

加号 (‘+’ U+002B) 是只能用于单个主题层级匹配的通配符。

在主题过滤器的任意层级都可以使用单层通配符，包括第一个和最后一个层级。然而它必须占据过滤器的整个层级 [MQTT-4.7.1-3]。可以在主题过滤器中的多个层级中使用它，也可以和多层通配符一起使用。

> 非规范评注
>
> 例如，“sport/tennis/+” 匹配 “sport/tennis/player1” 和 “sport/tennis/player2” ，但是不匹配 “sport/tennis/player1/ranking” 。同时，由于单层通配符只能匹配一个层级，“sport/+” 不匹配 “sport” 但是却匹配 “sport/”。
>
> 非规范评注

- “+” 是有效的。
- “+/tennis/#” 是有效的。
- “sport+” 是无效的。
- “sport/+/player1” 也是有效的。
- “/finance” 匹配 “+/+” 和 “/+” ，但是不匹配 “+”。

4.7.2 以$开头的主题 Topics beginning with $

服务端不能将 $ 字符开头的主题名匹配通配符 (#或+) 开头的主题过滤器 [MQTT-4.7.2-1]。服务端应该阻止客户端使用这种主题名与其它客户端交换消息。服务端实现可以将 $ 开头的主题名用作其他目的。

> 非规范评注

- $SYS/ 被广泛用作包含服务器特定信息或控制接口的主题的前缀。
- 应用不能使用 $ 字符开头的主题。

> 非规范评注

- 订阅 “#” 的客户端不会收到任何发布到以 “$” 开头主题的消息。
- 订阅 “+/monitor/Clients” 的客户端不会收到任何发布到 “$SYS/monitor/Clients” 的消息。
- 订阅 “$SYS/#” 的客户端会收到发布到以 “$SYS/” 开头主题的消息。
- 订阅 “$SYS/monitor/+” 的客户端会收到发布到 “$SYS/monitor/Clients” 主题的消息。
- 如果客户端想同时接受以 “$SYS/” 开头主题的消息和不以 $ 开头主题的消息，它需要同时订阅 “#” 和 ““$SYS/#”。

4.7.3 主题语义和用法 Topic semantic and usage

主题名和主题过滤器必须符合下列规则：

- 所有的主题名和主题过滤器必须至少包含一个字符 [MQTT-4.7.3-1]。
- 主题名和主题过滤器是区分大小写的。
- 主题名和主题过滤器可以包含空格。
- 主题名或主题过滤器以前置或后置斜杠 “/” 区分。
- 只包含斜杠 “/” 的主题名或主题过滤器是合法的。
- 主题名和主题过滤器不能包含空字符 (Unicode U+0000) [Unicode] [MQTT-4.7.3-2]。
- 主题名和主题过滤器是UTF-8编码字符串，它们不能超过65535字节 [MQTT-4.7.3-3]。见1.5.3节。

除了不能超过UTF-编码字符串的长度限制之外，主题名或主题过滤器的层级数量没有其它限制。

匹配订阅时，服务端不能对主题名或主题过滤器执行任何规范化（normalization）处理，不能修改或替换任何未识别的字符 [MQTT-4.7.3-4]。主题过滤器中的每个非通配符层级需要逐字符匹配主题名中对应的层级才算匹配成功。

> 非规范评注
>
> 使用UTF-8编码规则意味着，主题过滤器和主题名的比较可以通过比较编码后的UTF-8字节或解码后的Unicode字符。
>
> 非规范评注

- “ACCOUNTS” 和 “Accounts” 是不同的主题名。
- “Accounts payable” 是合法的主题名
- “/finance” 和 “finance” 是不同的。

如果订阅的主题过滤器与消息的主题名匹配，应用消息会被发送给每一个匹配的客户端订阅。主题可能是管理员在服务端预先定义好的，也可能是服务端收到第一个订阅或使用那个主题名的应用消息时动态添加的。服务端也可以使用一个安全组件有选择地授权客户端使用某个主题资源。

4.8 错误处理 Handling errors

除非另有说明，如果服务端或客户端遇到了协议违规的行为，它必须关闭传输这个协议违规控制报文的网络连接 [MQTT-4.8.0-1]。

客户端或服务端实现可能会遇到瞬时错误（Transient Error）（例如内部缓冲区满了的情况）导致无法成功处理MQTT报文。

如果客户端或服务端处理入站控制报文时遇到了瞬时错误，它必须关闭传输那个控制报文的网络连接 [MQTT-4.8.0-2]。如果服务端发现了瞬时错误，它不应该断开连接或者执行任何对其它客户端有影响的操作。

项目主页

- MQTT协议中文版

第五章 安全

目录

5.1 概述

本章的内容仅供参考，是非规范化的。然而，强烈推荐提供TLS的服务端实现应该使用TCP端口8883（IANA服务名：secure-mqtt）。

解决方案提供者需要考虑很多风险。例如：

- 设备可能会被盗用
- 客户端和服务端的静态数据可能是可访问的（可能会被修改）
- 协议行为可能有副作用（如计时器攻击）
- 拒绝服务攻击
- 通信可能会被拦截、修改、重定向或者泄露
- 虚假控制报文注入

MQTT方案通常部署在不安全的通信环境中。在这种情况下，协议实现通常需要提供这些机制：

- 用户和设备身份认证
- 服务端资源访问授权
- MQTT控制报文和内嵌应用数据的完整性校验
- MQTT控制报文和内嵌应用数据的隐私控制

作为传输层协议，MQTT仅关注消息传输，提供合适的安全功能是实现者的责任。使用TLS [RFC5246] 是比较普遍的选择。除了技术上的安全问题外，还有地理因素（例如美国欧盟安全港原则 [USEUSAFEHARB]），行业标准（例如第三方支付行业数据安全标准 [PCIDSS]），监管方面的考虑（例如萨班斯-奥克斯利法案 [SARBANES]）等问题。

5.2 MQTT解决方案：安全和认证

MQTT solutions: security and certification

协议实现可能想要符合特定的工业安全标准，如NIST网络安全框架 [NISTCSF] ，第三方支付行业数据安全标准 [PCIDSS] ，美国联邦信息处理标准 [FIPS1402] 和 NSA 加密组合B [NSAB] 。

在MQTT的补充出版物 （MQTT and the NIST Framework for Improving Critical Infrastructure Cybersecurity [MQTT NIST]） 中可以找到在NIST网络安全框架 [NISTCSF] 中使用MQTT的指导。使用行业证明、独立审计和认证技术有助于满足合规要求。

5.3 轻量级的加密与受限设备

Lightweight cryptography and constrained devices

广泛采用高级加密标准 [AES] 数据加密标准 [DES] 。

推荐使用为受限的低端设备特别优化过的轻量级加密国际标准 ISO 29192 [ISO29192] 。

5.4 实现注意事项 Implementation notes

实现和使用MQTT时需要考虑许多安全问题。下面的部分不应该被当作是一个 核对清单 。

协议实现可以实现下面的一部分或全部：

5.4.1 客户端身份验证 Authentication of Clients by the Server

CONNECT报文包含用户名和密码字段。实现可以决定如何使用这些字段的内容。实现者可以提供自己的身份验证机制，或者使用外部的认证系统如LDAP [RFC4511] 或OAuth [RFC6749] ，还可以利用操作系统的认证机制。

实现可以明文传递认证数据，混淆那些数据，或者不要求任何认证数据，但应该意识到这会增加中间人攻击和重放攻击的风险。5.4.5节介绍了确保数据私密的方法。

在客户端和服务端之间使用虚拟专用网（VPN）可以确保数据只被授权的客户端收到。

使用TLS [RFC5246] 时，服务端可以使用客户端发送的SSL证书验证客户端的身份。

实现可以允许客户端通过应用消息给服务端发送凭证用于身份验证。

5.4.2 客户端授权 Authorization of Clients by the Server

基于客户端提供的信息如用户名、客户端标识符（ClientId）、客户端的主机名或IP地址，或者身份认证的结果，服务端可以限制对某些服务端资源的访问。

5.4.3 服务端身份验证 Authentication of the Server by the Client

MQTT协议不是双向信任的，它没有提供客户端验证服务端身份的机制。

但是使用TLS [RFC5246] 时，客户端可以使用服务端发送的SSL证书验证服务端的身份。从单IP多域名提供MQTT服务的实现应该考虑RFC6066 [RFC6066] 第3节定义的TLS的SNI扩展。SNI允许客户端告诉服务端它要连接的服务端主机名。

实现可以允许服务端通过应用消息给客户端发送凭证用于身份验证。

在客户端和服务端之间使用虚拟专用网（VPN）可以确保客户端连接的是预期的服务器。

5.4.4 控制报文和应用消息的完整性 Integrity of Application Messages and Control Packets

应用可以在应用消息中单独包含哈希值。这样做可以为PUBLISH控制报文的网络传输和静态数据提供内容的完整性检查。

TLS [RFC5246] 提供了对网络传输的数据做完整性校验的哈希算法。

在客户端和服务端之间使用虚拟专用网（VPN）连接可以在VPN覆盖的网络段提供数据完整性检查。

5.4.5 控制报文和应用消息的保密性 Privacy of Application Messages and Control Packets

TLS [RFC5246] 可以对网络传输的数据加密。如果有效的TLS密码组合包含的加密算法为NULL，那么它不会加密数据。要确保客户端和服务端的保密，应避免使用这些密码组合。

应用可以单独加密应用消息的内容。这可以提供应用消息传输途中和静态数据的私密性。但不能给应用消息的其它属性如主题名加密。

客户端和服务端实现可以加密存储静态数据，例如可以将应用消息作为会话的一部分存储。

在客户端和服务端之间使用虚拟专用网（VPN）连接可以在VPN覆盖的网络段保证数据的私密性。

.5.4.6 消息传输的不可抵赖性 Non-repudiation of message transmission

应用设计者可能需要考虑适当的策略，以实现端到端的不可抵赖性（non-repudiation）。

5.4.7 检测客户端和服务端的盗用 Detecting compromise of Clients and Servers

使用TLS [RFC5246] 的客户端和服务端实现应该能够确保，初始化TLS [RFC5246] 连接时提供的SSL证书是与主机名（客户端要连接的或服务端将被连接的）关联的。

使用TLS [RFC5246] 的客户端和服务端实现，可以选择提供检查证书吊销列表 (CRLs [RFC5280]) 和在线证书状态协议 (OSCP) [RFC6960] 的功能，拒绝使用被吊销的证书。

物理部署可以将防篡改硬件与应用消息的特殊数据传输结合。例如，一个仪表可能会内置一个GPS以确保没有在未授权的地区使用。IEEE安全设备认证 [IEEE 802.1AR] 就是用于实现这个机制的一个标准，它使用加密绑定标识符验证设备身份。

5.4.8 检测异常行为 Detecting abnormal behaviors

服务端实现可以监视客户端的行为，检测潜在的安全风险。例如：

- 重复的连接请求
- 重复的身份验证请求
- 连接的异常终止
- 主题扫描（请求发送或订阅大量主题）
- 发送无法送达的消息（没有订阅者的主题）
- 客户端连接但是不发送数据

发现违反安全规则的行为，服务端实现可以断开客户端连接。

服务端实现检测不受欢迎的行为，可以基于IP地址或客户端标识符实现一个动态黑名单列表。

服务部署可以使用网络层次控制（如果可用）实现基于IP地址或其它信息的速率限制或黑名单。

5.4.9 其它的安全注意事项 Other security considerations

如果客户端或服务端的SSL证书丢失，或者我们考虑证书被盗用或者被吊销(利用 CRLs [RFC5280] 和 OSCP [RFC6960])的情况。

客户端或服务端验证凭证时，如果发现用户名和密码丢失或被盗用，应该吊销或者重新发放。

在使用长连接时：

- 客户端和服务端使用TLS [RFC5246] 时应该允许重新协商会话以确认新的加密参数（替

换会话密钥，更换密码组合，更换认证凭证）。
- 服务端可以断开客户端连接，并要求他们使用新的凭证重新验证身份。

受限网络上的受限设备和客户端可以使用TLS会话恢复 [RFC5077] 降低TLS会话重连 [RFC5246] 的成本。

连接到服务端的客户端与其它连接到服务端的客户端 之间有一个信任传递关系，它们都有权在同一个主题上发布消息。

5.4.10 使用SOCKS代理 Use of SOCKS

客户端实现应该意识到某些环境要求使用SOCKSv5 [RFC1928] 代理创建出站的网络连接。某些MQTT实现可以利用安全隧道（如SSH）通过SOCKS代理。一个实现决定支持SOCKS时，它们应该同时支持匿名的和用户名密码验证的SOCKS代理。对于后一种情况，实现应该意识到SOCKS可能使用明文认证，因此应该避免使用相同的凭证连接MQTT服务器。

5.4.11 安全配置文件 Security profiles

实现者和方案设计者可能希望将安全当作配置文件集合应用到MQTT协议中。下面描述的是一个分层的安全等级结构。

开放通信配置

使用开放通信配置时，MQTT协议运行在一个没有内置额外安全通信机制的开放网络上。

安全网络通信配置

使用安全网络通信配置时，MQTT协议运行在有安全控制的物理或虚拟网络上，如VPN或物理安全网络。

安全传输配置

使用安全传输配置时，MQTT协议运行在使用TLS [RFC5246] 的物理或虚拟网络上，它提供了身份认证，完整性和保密性。

使用内置的用户名和密码字段，TLS [RFC5246] 客户端身份认证可被用于（或者代替）MQTT客户端认证。

工业标准的安全配置

可以预料的是，MQTT被设计为支持很多工业标准的应用配置，每一种定义一个威胁模型和用于定位威胁的特殊安全机制。特殊的安全机制推荐从下面的方案中选择：

[NISTCSF] NIST网络安全框架 [NIST7628] NISTIR 7628智能电网网络安全指南 [FIPS1402] (FIPS PUB 140-2) 加密模块的安全要求 [PCIDSS] PCI-DSS第三方支付行业数据安全标准 [NSAB] NSA加密组合B

项目主页

- MQTT协议中文版

第六章 使用WebSocket作为网络层

目录

如果MQTT在WebSocket [RFC6455] 连接上传输，必须满足下面的条件：

- MQTT控制报文必须使用WebSocket二进制数据帧发送。如果收到任何其它类型的数据帧，接收者必须关闭网络连接 [MQTT-6.0.0-1]。
- 单个WebSocket数据帧可以包含多个或者部分MQTT报文。接收者不能假设MQTT控制报文按WebSocket帧边界对齐 [MQTT-6.0.0-2]。
- 客户端必须将字符串 **mqtt** 包含在它提供的WebSocket子协议列表里 [MQTT-6.0.0-3]。
- 服务端选择和返回的WebSocket子协议名必须是 **mqtt** [MQTT-6.0.0-4] 。
- 用于连接客户端和服务器的WebSocket URI对MQTT协议没有任何影响。

6.1 IANA注意事项 IANA Considerations

> 本规范请求IANA在WebSocket子协议名条目下注册WebSocket MQTT子协议，使用下列数据：

图例 **6.1 - IANA WebSocket**标识符

子协议标识符	**mqtt**
子协议通用名	mqtt
子协议定义	http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

项目主页

- MQTT协议中文版

第七章 一致性 Conformance

目录

MQTT规范定义了MQTT客户端实现和MQTT服务端实现的一致性要求

MQTT实现可以同时是MQTT客户端和MQTT服务端。接受入站连接和建立到其它服务端的出站连接的服务端必须同时符合MQTT客户端和MQTT服务端的要求 [MQTT-7.0.0-1]。

为了与任何其它的一致性实现交互操作，一致性实现不能要求使用在本规范之外定义的任何扩展 [MQTT-7.0.0-2]。

7.1 一致性目标 Conformance Targets

7.1.1 MQTT服务端 MQTT Server

一个MQTT服务端只有满足下面所有的要求才算是符合本规范：

1. 服务端发送的所有控制报文的格式必须符合第二章和第三章描述的格式
2. 遵守第4.7节描述的主题匹配规则。
3. 满足下列章节中所有必须级别的要求，明确仅适用于对客户端的除外：
    - 第一章 – 介绍
    - 第二章 – MQTT控制报文格式
    - 第三章 – MQTT控制报文
    - 第四章 – 操作行为
    - 第六章 – （如果MQTT的网络层是WebSocket）
    - 第七章 – 一致性目标

满足一致性要求的服务端必须支持使用一个或多个底层传输协议，只要它提供有序的、可靠的、双向字节流（从客户端到服务端和从服务端到客户端）[MQTT-7.1.1-1]。但是一致性并不依赖于它支持任何特定的传输协议。服务端可以支持第4.2节列出的任何传输协议，或者任何其它满足 [MQTT-7.1.1-1] 要求的传输协议。

7.1.2 MQTT客户端 MQTT Client

一个MQTT客户端只有满足下面所有的要求才算是符合本规范：

1. 客户端发送的所有控制报文的格式必须符合第二章和第三章描述的格式
2. 满足下列章节中所有必须级别的要求，明确仅适用于对服务端的除外：
    - 第一章 – 介绍
    - 第二章 – MQTT控制报文格式
    - 第三章 – MQTT控制报文
    - 第四章 – 操作行为
    - 第六章 – （如果MQTT的网络层是WebSocket）
    - 第七章 – 一致性目标

满足一致性要求的客户端必须支持使用一个或多个底层传输协议，只要它提供有序的、可靠的、双向字节流（从客户端到服务端和从服务端到客户端）[MQTT-7.1.2-1]。但是一致性并不依赖于它支持任何特定的传输协议。客户端可以支持第4.2节列出的任何传输协议，或者任何其它满足 [MQTT-7.1.2-1] 要求的传输协议。

项目主页

- MQTT协议中文版

附录B 强制性规范声明（非规范）

目录

这个附录是非规范的，只作为本文档正文中可以找到的大量一致性声明的摘要提供。一致性要求的限制列表见第七章。

表格：强制性规范声明

声明序号	规范声明
[MQTT-1.5.3-1]	UTF-8编码字符串中的字符数据必须是按照Unicode规范 [Unicode] 定义的和在R [RFC3629] 中重申的有效的UTF-8格式。特别需要指出的是，这些数据不能包含 U+D800和U+DFFF之间的数据。如果服务端或客户端收到了一个包含无效UTF-8 报文，它必须关闭网络连接。
[MQTT-1.5.3-2]	UTF-8编码的字符串不能包含空字符U+0000。如果客户端或服务端收到了一个包 的控制报文，它必须关闭网络连接。
[MQTT-1.5.3-3]	UTF-8编码序列0XEF 0xBB 0xBF总是被解释为U+FEFF（零宽度非换行空白字符 出现在字符串的什么位置，报文接收者都不能跳过或者剥离它。
[MQTT-2.2.2-1]	表格 2.2中任何标记为“保留”的标志位，都是保留给以后使用的，必须设置为表格 值。
[MQTT-2.2.2-2]	如果收到非法的标志，接收者必须关闭网络连接。
[MQTT-2.3.1-1]	SUBSCRIBE，UNSUBSCRIBE和PUBLISH（QoS大于0）控制报文必须包含一 位报文标识符（Packet Identifier。
[MQTT-2.3.1-2]	客户端每次发送一个新的这些类型的报文时都必须分配一个当前未使用的报文标
[MQTT-2.3.1-3]	如果一个客户端要重发这个特殊的控制报文，在随后重发那个报文时，它必须使 识符。当客户端处理完这个报文对应的确认后，这个报文标识符就释放可重用。 PUBLISH对应的是PUBACK，QoS 2的PUBLISH对应的是PUBCOMP，与SUBS

	UNSUBSCRIBE对应的分别是SUBACK或UNSUBACK
[MQTT-2.3.1-4]	发送一个QoS 0的PUBLISH报文时，相同的条件也适用于服务端。
[MQTT-2.3.1-5]	QoS设置为0的PUBLISH报文不能包含报文标识符。
[MQTT-2.3.1-6]	PUBACK, PUBREC, PUBREL报文必须包含与最初发送的PUBLISH报文相同的报 符。
[MQTT-2.3.1-7]	与 [MQTT-2.3.1-6] 类似，SUBACK和UNSUBACK必须包含在对应的SUBSCRIB UNSUBSCRIBE报文中使用的报文标识符。
[MQTT-3.1.0-1]	客户端到服务端的网络连接建立后，客户端发送给服务端的第一个报文必须是CO 文。
[MQTT-3.1.0-2]	在一个网络连接上，客户端只能发送一次CONNECT报文。服务端必须将客户端 个CONNECT报文当作协议违规处理并断开客户端的连接。
[MQTT-3.1.2-1]	如果协议名不正确服务端可以断开客户端的连接，也可以按照某些其它规范继续 CONNECT报文。对于后一种情况，按照本规范，服务端不能继续处理CONNEC
[MQTT-3.1.2-2]	如果发现不支持的协议级别，服务端必须给发送一个返回码为0x01（不支持的协 CONNACK报文响应CONNECT报文，然后断开客户端的连接。
[MQTT-3.1.2-3]	服务端必须验证CONNECT控制报文的保留标志位（第0位）是否为0，如果不为 客户端连接。
[MQTT-3.1.2-4]	如果清理会话（CleanSession）标志被设置为0，服务端必须基于当前会话（使 识符识别）的状态恢复与客户端的通信。如果没有与这个客户端标识符关联的会 必须创建一个新的会话。在连接断开之后，当连接断开后，客户端和服务端必须 息。
[MQTT-3.1.2-5]	当清理会话标志为0的会话连接断开之后，服务端必须将之后的QoS 1和QoS 2级 存为会话状态的一部分，如果这些消息匹配断开连接时客户端的任何订阅。
[MQTT-3.1.2-6]	如果清理会话（CleanSession）标志被设置为1，客户端和服务端必须丢弃之前 并开始一个新的会话。会话仅持续和网络连接同样长的时间。与这个会话关联的 能被任何之后的会话重用。
[MQTT-3.1.2.7]	保留消息不是服务端会话状态的一部分，会话终止时不能删除保留消息。
[MQTT-3.1.2-8]	遗嘱标志（Will Flag）被设置为1，表示如果连接请求被接受了，遗嘱（Will Mes 必须被存储在服务端并且与这个网络连接关联。之后网络连接关闭时，服务端必 遗嘱消息，除非服务端收到DISCONNECT报文时删除了这个遗嘱消息。
[MQTT-3.1.2-9]	如果遗嘱标志被设置为1，连接标志中的Will QoS和Will Retain字段会被服务端用 效载荷中必须包含Will Topic和Will Message字段。
[MQTT-3.1.2-10]	一旦被发布或者服务端收到了客户端发送的DISCONNECT报文，遗嘱消息就必 会话状态中移除。
[MQTT-3.1.2-11]	如果遗嘱标志被设置为0，连接标志中的Will QoS和Will Retain字段必须设置为0 载荷中不能包含Will Topic和Will Message字段。

[MQTT-3.1.2-12]	如果遗嘱标志被设置为0，网络连接断开时，不能发送遗嘱消息。.
[MQTT-3.1.2-13]	如果遗嘱标志被设置为0，遗嘱QoS也必须设置为0(0x00)。
[MQTT-3.1.2-14]	如果遗嘱标志被设置为1，遗嘱QoS的值可以等于0(0x00)，1(0x01)，2(0x02)。‘ 等于3。
[MQTT-3.1.2-15]	如果遗嘱标志被设置为0，遗嘱保留（Will Retain）标志也必须设置为0。
[MQTT-3.1.2-16]	如果遗嘱保留被设置为0，服务端必须将遗嘱消息当作非保留消息发布。
[MQTT-3.1.2-17]	如果遗嘱保留被设置为1，服务端必须将遗嘱消息当作保留消息发布。
[MQTT-3.1.2-18]	如果用户名（User Name）标志被设置为0，有效载荷中不能包含用户名字段。
[MQTT-3.1.2-19]	如果用户名（User Name）标志被设置为1，有效载荷中必须包含用户名字段。
[MQTT-3.1.2-20]	如果密码（Password）标志被设置为0，有效载荷中不能包含密码字段。
[MQTT-3.1.2-21]	如果密码（Password）标志被设置为1，有效载荷中必须包含密码字段
[MQTT-3.1.2-22]	如果用户名标志被设置为0，密码标志也必须设置为0。
[MQTT-3.1.2-23]	客户端负责保证控制报文发送的时间间隔不超过保持连接的值。如果没有任何其 文可以发送，客户端必须发送一个PINGREQ报文。
[MQTT-3.1.2-24]	如果保持连接的值非零，并且服务端在一点五倍的保持连接时间内没有收到客户 文，它必须断开客户端的网络连接，认为网络连接已断开。
[MQTT-3.1.3-1]	如果包含的话，必须按这个顺序出现：客户端标识符，遗嘱主题，遗嘱消息，用 码。
[MQTT-	服务端使用客户端标识符 (ClientId) 识别客户端。连接服务端的每个客户端都有 端标识符（ClientId）。客户端和服务端都必须使用ClientId识别两者之间的MQT

	的状态。
[MQTT-3.1.3-3]	客户端标识符 (ClientId) 必须存在而且必须是CONNECT报文有效载荷的第一个
[MQTT-3.1.3-4]	客户端标识符必须是1.5.3节定义的UTF-8编码字符串。
[MQTT-3.1.3-5]	服务端必须允许1到23个字节长的UTF-8编码的客户端标识符，客户端标识符只 符："0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTU 写字母，小写字母和数字）。
[MQTT-3.1.3-6]	服务端可以允许客户端提供一个零字节的客户端标识符 (ClientId) ，如果这样做 必须将这看作特殊情况并分配唯一的客户端标识符给那个客户端。然后它必须假 供了那个唯一的客户端标识符，正常处理这个CONNECT报文。
[MQTT-3.1.3-7]	如果客户端提供了一个零字节的客户端标识符，它必须同时将清理会话标志设置
[MQTT-3.1.3-8]	如果客户端提供的ClientId为零字节且清理会话标志为0，服务端必须发送返回码 示标识符不合格）的CONNACK报文响应客户端的CONNECT报文，然后关闭网
[MQTT-3.1.3-9]	如果服务端拒绝了这个ClientId，它必须发送返回码为0x02（表示标识符不合格） CONNACK报文响应客户端的CONNECT报文，然后关闭网络连接。
[MQTT-3.1.3-10]	遗嘱主题必须是 1.5.3节定义的UTF-8编码字符串。
[MQTT-3.1.3-11]	用户名必须是 1.5.3节定义的UTF-8编码字符串。
[MQTT-3.1.4-1]	服务端必须按照3.1节的要求验证CONNECT报文，如果报文不符合规范，服务 CONNACK报文直接关闭网络连接。
[MQTT-3.1.4-2]	如果ClientId表明客户端已经连接到这个服务端，那么服务端必须断开原有的客
[MQTT-3.1.4-3]	服务端必须按照 3.1.2.4节的描述执行清理会话的过程。
[MQTT-3.1.4-4]	服务端必须发送返回码为零的CONNACK报文作为CONNECT报文的确认响应。
[MQTT-3.1.4-5]	如果服务端拒绝了CONNECT，它不能处理客户端在CONNECT报文之后发送的
[MQTT-3.2.0-1]	服务端发送CONNACK报文响应从客户端收到的CONNECT报文。服务端发送给 一个报文必须是CONNACK。
[MQTT-3.2.2-1]	如果服务端收到清理会话（CleanSession）标志为1的连接，除了将CONNACK 回码设置为0之外，还必须将CONNACK报文中的当前会话设置（Session Prese 0。
[MQTT-3.2.2-2]	如果服务端收到一个CleanSession为0的连接，当前会话标志的值取决于服务端 存了ClientId对应客户端的会话状态。如果服务端已经保存了会话状态，它必须 报文中的当前会话标志设置为1。

	报文中的当前会话标志设置为1。
[MQTT-3.2.2-3]	如果服务端没有已保存的会话状态，它必须将CONNACK报文中的当前会话设置 将CONNACK报文中的返回码设置为0。
[MQTT-3.2.2-4]	如果服务端发送了一个包含非零返回码的CONNACK报文，它必须将当前会话标
[MQTT-3.2.2-5]	如果服务端发送了一个包含非零返回码的CONNACK报文，那么它必须关闭网络
[MQTT-3.2.2-6]	如果认为上表格3.1中的所有连接返回码都不太合适，那么服务端必须关闭网络 发送CONNACK报文。
[MQTT-3.3.1-1]	客户端或服务端请求重发一个PUBLISH报文时，必须将DUP标志设置为1。
[MQTT-3.3.1-2]	对于QoS 0的消息，DUP标志必须设置为0
[MQTT-3.3.1-3]	服务端发送PUBLISH报文给订阅者时，收到（入站）的PUBLISH报文的DUP标 被传播。发送（出站）的PUBLISH报文与收到（入站）的PUBLISH报文中的DU 立设置的，它的值必须单独的根据发送（出站）的PUBLISH报文是否是一个重发
[MQTT-3.3.1-4]	PUBLISH报文不能将QoS所有的位设置为1。如果服务端或客户端收到QoS所有 PUBLISH报文，它必须关闭网络连接。
[MQTT-3.3.1-5]	如果客户端发给服务端的PUBLISH报文的保留（RETAIN）标志被设置为1，服务 这个应用消息和它的服务质量等级（QoS），以便它可以被分发给未来的主题名 者。
[MQTT-3.3.1-6]	一个新的订阅建立时，对每个匹配的主题名，如果存在最近保留的消息，它必须 个订阅者。
[MQTT-3.3.1-7]	如果服务端收到一条保留（RETAIN）标志为1的QoS 0消息，它必须丢弃之前为 留的任何消息。它应该将这个新的QoS 0消息当作那个主题的新保留消息，但是 可以选择丢弃它 — 如果这种情况发生了，那个主题将没有保留消息。
[MQTT-3.3.1-8]	服务端发送PUBLISH报文给客户端时，如果消息是作为客户端一个新订阅的结果 须将报文的保留标志设为1。
[MQTT-3.3.1-9]	当一个PUBLISH报文发送给客户端是因为匹配一个已建立的订阅时，服务端必须 设为0，不管它收到的这个消息中保留标志的值是多少。
[MQTT-3.3.1-10]	保留标志为1且有效载荷为零字节的PUBLISH报文会被服务端当作正常消息处理 送给订阅主题匹配的客户端。此外，同一个主题下任何现存的保留消息必须被移 个主题之后的任何订阅者都不会收到一个保留消息。
[MQTT-3.3.1-11]	服务端不能存储零字节的保留消息。
[MQTT-3.3.1-12]	如果客户端发给服务端的PUBLISH报文的保留标志位0，服务端不能存储这个消 除或替换任何现存的保留消息。
[MQTT-3.3.2-1]	主题名必须是PUBLISH报文可变报头的第一个字段。它必须是 1.5.3节定义的UT 字符串。

[MQTT-3.3.2-2]	PUBLISH报文中的主题名不能包含通配符。
[MQTT-3.3.2-3]	服务端发送给订阅客户端的PUBLISH报文的主题名必须匹配该订阅的主题过滤器 节定义的匹配过程）。
[MQTT-3.3.4-1]	PUBLISH报文的接收者必须按照根据PUBLISH报文中的QoS等级发送响应，见 述。
[MQTT-3.3.5-1]	服务端必须将消息分发给所有订阅匹配的QoS等级最高的客户端。
[MQTT-3.3.5-2]	如果服务端实现不授权某个客户端发布PUBLISH报文，它没有办法通知那个客户 按照正常的QoS规则发送一个正面的确认，或者关闭网络连接。
[MQTT-3.6.1-1]	PUBREL控制报文固定报头的第3,2,1,0位是保留位，必须被设置为0,0,1,0。服务 它的任何值都当做是不合法的并关闭网络连接。
[MQTT-3.8.1-1]	SUBSCRIBE控制报固定报头的第3,2,1,0位是保留位，必须分别设置为0,0,1,0。 将其它的任何值都当做是不合法的并关闭网络连接。
[MQTT-3.8.3-1]	SUBSCRIBE报文有效载荷中的主题过滤器列表必须是1.5.3节定义的UTF-8字符
[MQTT-3.8.3-2]	如果服务端选择不支持包含通配符的主题过滤器，必须拒绝任何包含通配符过滤 求。
[MQTT-3.8.3-3]	SUBSCRIBE报文的有效载荷必须包含至少一对主题过滤器 和 QoS等级字段组合 载荷的SUBSCRIBE报文是违反协议的。
[MQTT-3-8.3-4]	如果有效载荷中的任何位是非零值，或者QoS不等于0,1或2，服务端必须认为SU 报文是不合法的并关闭网络连接。
[MQTT-3.8.4-1]	服务端收到客户端发送的一个SUBSCRIBE报文时，必须使用SUBACK报文响应
[MQTT-3.8.4-2]	SUBACK报文必须和等待确认的SUBSCRIBE报文有相同的报文标识符。
[MQTT-3.8.4-3]	如果服务端收到一个SUBSCRIBE报文，报文的主题过滤器与一个现存订阅的主 同，那么必须使用新的订阅彻底替换现存的订阅。新订阅的主题过滤器和之前订 但是它的最大QoS值可以不同。与这个主题过滤器匹配的任何现存的保留消息必 但是发布流程不能中断。
[MQTT-3.8.4-4]	如果服务端收到包含多个主题过滤器的SUBSCRIBE报文，它必须如同收到了一 SUBSCRIBE报文一样处理那个，除了需要将它们的响应合并到一个单独的SUB 送。
[MQTT-3.8.4-5]	服务端发送给客户端的SUBACK报文对每一对主题过滤器 和QoS等级都必须包 码。这个返回码必须表示那个订阅被授予的最大QoS等级，或者表示这个订阅失
[MQTT-3.8.4-6]	服务端可以授予比订阅者要求的低一些的QoS等级。为响应订阅而发出的消息的 QoS必须是原始发布消息的QoS和服务端授予的QoS两者中的最小值。如果原始 是1而被授予的最大QoS是0，允许服务端重复发送一个消息的副本给订阅者。
[MQTT-3.9.3-1]	返回码的顺序必须和SUBSCRIBE报文中主题过滤器的顺序相同。

3.9.3-1]	
[MQTT-3.9.3-2]	0x00, 0x01, 0x02, 0x80之外的SUBACK返回码是保留的，不能使用。
[MQTT-3.10.1-1]	UNSUBSCRIBE报文固定报头的第3,2,1,0位是保留位且必须分别设置为0,0,1,0。认为任何其它的值都是不合法的并关闭网络连接。
[MQTT-3.10.3-1]	UNSUBSCRIBE报文中的主题过滤器必须是连续打包的、按照1.5.3节定义的UTI串。
[MQTT-3.10.3-2]	UNSUBSCRIBE报文的有效载荷必须至少包含一个消息过滤器。没有有效载荷的UNSUBSCRIBE报文是违反协议的。
[MQTT-3.10.4-1]	UNSUBSCRIBE报文提供的主题过滤器（无论是否包含通配符）必须与服务端持户端的当前主题过滤器集合逐个字符比较。如果有任何过滤器完全匹配，那么它自己的订阅将被删除，否则不会有进一步的处理。
[MQTT-3.10.4-2]	如果服务端删除了一个订阅，它必须停止分发任何新消息给这个客户端。
[MQTT-3.10.4-3]	如果服务端删除了一个订阅，它必须完成分发任何已经开始往客户端发送的QoS的消息。
[MQTT-3.10.4-4]	服务端必须发送UNSUBACK报文响应客户端的UNSUBSCRIBE请求。UNSUBA包含和UNSUBSCRIBE报文相同的报文标识符。
[MQTT-3.10.4-5]	即使没有删除任何主题订阅，服务端也必须发送一个SUBACK响应。
[MQTT-3.10.4-6]	如果服务端收到包含多个主题过滤器的UNSUBSCRIBE报文，它必须如同收到了个UNSUBSCRIBE报文一样处理那个报文，除了将它们的响应合并到一个单独的UNSUBACK报文外。
[MQTT-3.12.4-1]	服务端必须发送 PINGRESP报文响应客户端的PINGREQ报文。
[MQTT-3.14.1-1]	服务端必须验证所有的保留位都被设置为0，如果它们不为0必须断开连接。
[MQTT-3.14.4-1]	客户端发送DISCONNECT报文之后，必须关闭网络连接。
[MQTT-3.14.4-2]	客户端发送DISCONNECT报文之后，不能通过那个网络连接再发送任何控制报
[MQTT-	

3]	体描述见 3.1.2.5节。
[MQTT-4.1.0-1]	在整个会话期间，客户端和服务端都必须存储会话状态。
[MQTT-4.1.0-2]	会话必须至少持续和它的活跃网络连接同样长的时间。
[MQTT-4.3.1-1]	对于QoS 0的分发协议，发送者必须发送QoS等于0，DUP等于0的PUBLISH报文
[MQTT-4.4.0-1]	客户端设置清理会话（CleanSession）标志为0重连时，客户端和服务端必须使文标识符重发任何未确认的PUBLISH报文（如果QoS>0）和PUBREL报文。
[MQTT-4.5.0-1]	服务端接管入站应用消息的所有权时，它必须将消息添加到订阅匹配的客户端的中。匹配规则定义见 4.7节。
[MQTT-4.5.0-2]	客户端必须按照可用的服务质量（QoS）规则确认它收到的任何PUBLISH报文，是否处理报文包含的应用消息。
[MQTT-4.6.0-1]	重发任何之前的PUBLISH报文时，必须按原始PUBLISH报文的发送顺序重发（和QoS 2消息）。
[MQTT-4.6.0-2]	必须按照对应的PUBLISH报文的顺序发送PUBACK报文（QoS 1消息）。
[MQTT-4.6.0-3]	必须按照对应的PUBLISH报文的顺序发送PUBREC报文（QoS 2消息）。
[MQTT-4.6.0-4]	必须按照对应的PUBREC报文的顺序发送PUBREL报文（QoS 2消息）。
[MQTT-4.6.0-5]	服务端必须默认认为每个主题都是有序的。它可以提供一个管理功能或其它机制一个或多个主题当作是无序的。
[MQTT-4.6.0-6]	服务端处理发送给有序主题的消息时，必须按照上面的规则将消息分发给每个订外，它必须按照从客户端收到的顺序发送PUBLISH报文给消费者（对相同的主题
[MQTT-4.7.1-1]	主题过滤器中可以使用通配符，但是主题名不能使用通配符。
[MQTT-4.7.1-2]	多层通配符必须位于它自己的层级或者跟在主题层级分隔符后面。不管哪种情况是主题过滤器的最后一个字符。
[MQTT-4.7.1-3]	在主题过滤器的任意层级都可以使用单层通配符，包括第一个和最后一个层级。占据过滤器的整个层级。
[MQTT-4.7.2-1]	服务端不能将 $ 字符开头的主题名匹配通配符 (#或+) 开头的主题过滤器。
[MQTT-4.7.3-1]	所有的主题名和主题过滤器必须至少包含一个字符。
[MQTT-4.7.3-2]	主题名和主题过滤器不能包含空字符 (Unicode U+0000) [Unicode] 。
[MQTT-4.7.3-3]	主题名和主题过滤器是UTF-8编码字符串，它们不能超过65535字节。

[MQTT-4.7.3-4]	匹配订阅时，服务端不能对主题名或主题过滤器执行任何规范化（normalization 能修改或替换任何未识别的字符。
[MQTT-4.8.0-1]	除非另有说明，如果服务端或客户端遇到了协议违规的行为，它必须关闭传输这 控制报文的网络连接。
[MQTT-4.8.0-2]	如果客户端或服务端处理入站控制报文时遇到了瞬时错误，它必须关闭传输那个 网络连接。
[MQTT-6.0.0-1]	MQTT控制报文必须使用WebSocket二进制数据帧发送。如果收到任何其它类型 接收者必须关闭网络连接。
[MQTT-6.0.0-2]	单个WebSocket数据帧可以包含多个或者部分MQTT报文。接收者不能假设MQT 按WebSocket帧边界对齐。
[MQTT-6.0.0-3]	客户端必须将字符串 **mqtt** 包含在它提供的WebSocket子协议列表里。
[MQTT-6.0.0-4]	服务端选择和返回的WebSocket子协议名必须是 **mqtt**
[MQTT-7.0.0-1]	MQTT实现可以同时是MQTT客户端和MQTT服务端。接受入站连接和建立到其它 站连接的服务端必须同时符合MQTT客户端和MQTT服务端的要求。
[MQTT-7.0.0-2]	为了与任何其它的一致性实现交互操作，一致性实现不能要求使用在本规范之外 扩展。
[MQTT-7.1.1-1]	满足一致性要求的服务端必须支持使用一个或多个底层传输协议，只要它提供有 的、双向字节流（从客户端到服务端和从服务端到客户端）
[MQTT-7.1.2-1]	满足一致性要求的客户端必须支持使用一个或多个底层传输协议，只要它提供有 的、双向字节流（从客户端到服务端和从服务端到客户端）

关于QoS的补充说明

[MQTT-4.3.2-1] 对于QoS 1的分发协议，发送者

- 每次发送新的应用消息都必须分配一个未使用的报文标识符。
- MUST send a PUBLISH Packet containing this Packet Identifier with QoS=1, DUP=0.
- 发送的PUBLISH报文必须包含报文标识符且QoS等于1，DUP等于0。
- 必须将这个PUBLISH报文看作是 未确认的 ，直到从接收者那收到对应的PUBACK报文。4.4节有一个关于未确认消息的讨论。

[MQTT-4.3.2-2] 对于QoS 1的分发协议，接收者

- 响应的PUBACK报文必须包含一个报文标识符，这个标识符来自接收到的、已经接受所有权的PUBLISH报文。
- 发送了PUBACK报文之后，接收者必须将任何包含相同报文标识符的入站PUBLISH报文当作一个新的消息，并忽略它的DUP标志的值。

[MQTT-4.3.3-1] 对于QoS 2的分发协议，发送者

- 必须给要发送的新应用消息分配一个未使用的报文标识符。
    - MUST send a PUBLISH packet containing this Packet Identifier with QoS=2, DUP=0.
- 发送的PUBLISH报文必须包含报文标识符且报文的QoS等于2,，DUP等于0。
    - 必须将这个PUBLISH报文看作是 未确认的 ，直到从接收者那收到对应的PUBREC报文。4.4节有一个关于未确认消息的讨论。
    - 收到PUBREC报文后必须发送一个PUBREL报文。PUBREL报文必须包含与原始PUBLISH报文相同的报文标识符。
    - 必须将这个PUBREL报文看作是 未确认的 ，直到从接收者那收到对应的PUBCOMP报文。
    - 一旦发送了对应的PUBREL报文就不能重发这个PUBLISH报文。

[MQTT-4.3.3-2] 对于QoS 2的分发协议，接收者

- 响应的PUBREC报文必须包含报文标识符，这个标识符来自接收到的、已经接受所有权的PUBLISH报文。
- 在收到对应的PUBREL报文之前，接收者必须发送PUBREC报文确认任何后续的具有相同标识符的PUBLISH报文。 在这种情况下，它不能重复分发消息给任何后续的接收者。
- 响应PUBREL报文的PUBCOMP报文必须包含与PUBREL报文相同的标识符。
    - 发送PUBCOMP报文之后，接收者必须将包含相同报文标识符的任何后续PUBLISH报文当作一个新的发布。

项目主页

- MQTT协议中文版